Deep Learning in Real Estate Prediction: An Empirical Study on California House Prices

Audrey Chen

Received April 06, 2024
Accepted July 29, 2024
Electronic access September 15, 2024

Machine learning has become increasingly prevalent in the real estate industry for predicting house values and providing key indicators for making informed market trend decisions. In this research, by integrating computer science and artificial intelligence (AI), various machine learning (ML) algorithms and data features were tested to determine which procedures yielded the highest accuracy. These features will assist real estate investors with predictive analytics for pricing trends, better market segmentations, risk assessments, demand forecasting, and portfolio optimization. This process can also be applied and tested for Airbnb investors in the short-term rental (STR) market in the current dynamic real estate market. The imported dataset encompasses variables from the California housing market, layered with additional geographic parameters. Linear regression, support vector machines with linear, polynomial, and radial basis function kernels, and deep neural networks were reviewed and tested. To assess the accuracy of the regression models, RMSE was used as an evaluation metric. Upon analyzing the RMSE results, the SVR with the RBF kernel exhibited the lowest error values compared to the performance of other models. This indicates the suitability of this model for the regression task, highlighting the impact of model hyperparameter choice on task performance. The exceptional performance of the RBF kernel makes it a valuable candidate for real-world applications and future house price predictions in the volatile real estate market. These findings lay the foundation for model optimization, feature engineering, and further investigation into the dataset's characteristics and alternative model architectures within the larger real estate market.

Keywords: Machine learning; Deep Neural Network; Support Vector Machine; Linear regression; Real Estate

Introduction

The real estate market of the United States stands as a cornerstone of the nation's economy, playing a significant role in citizens' day-to-day lives and investments. Whether a homeowner, renter, or business owner, the real estate market and its state directly influence one's life and economic condition. Owning a home is not just a financial investment - it also fosters stability, security, and a home for Americans. Real estate assets dominate a substantial portion of an individual's net worth. Thus, the real estate market is a critical driver of wealth nationwide [1]. Through a larger lens, the market is central to job creation and employment, serving as a primary pillar in the financial sector and an assessor of the nation's economic health. The U.S. real estate market has an extensive history of peaks and lows among uncertain volatility. The financial crisis of 2007-08 is an example of one of the most transformative incidents, which altered the 21st-century financial market and left a lasting footprint on the industry [2]. Investing in real estate investment properties serves multiple purposes and constitutes income generation, hedging against inflation, associated tax benefits, and appreciation of property values over time.

Traditional housing price analysis methods in the real estate market often comprises of fundamental and technical analyses. Fundamental analysis involves examining various factors, such as financial news, market reports, and economic articles, to better understand how the real estate market operates in a broader economic context. Future trend predictions and economic health observations can be made by considering employment, interest rates, and consumer sentiment [3]. Moreover, fundamental analysis often consists of insider information and industry-specific knowledge used by specialized analysts to provide insight into particular housing markets or properties. On the contrary, more advanced analysis relies on historical market data, typically covering the past few years. Financial analysts survey price charts, transaction volumes, and a variety of technical indicators to identify particular patterns and trends throughout the real estate market. Any sustained price trends and consumer data over the years can provide valuable guidance for investors and buyers [3]. Applying advanced analysis and recognizing distinct trends often aid investors in gaining insight into potential future economic movements and making more informed property investment decisions.

Analyzing the real estate market and housing prices using traditional methods can be challenging due to several factors. Markets are highly localized, and certain data that apply to one area

may not be relevant to another[4]. Traditional methods often rely on factors of the broader economic market, which lessens the likelihood of capturing the distinctions between specific housing markets. Additionally, numerous variables influence house prices, including supply and demand, neighborhood attributes, and property conditions, making it difficult to generalize the market amongst traditional procedures[4]. The real estate market is also highly sensitive to various external factors, including government policies, interest rates, and global economic conditions. This further complicates the analysis of price trends. Another possible challenge is obtaining accurate and up-to-date data, as certain real estate transactions are not always publicly available or disclosed in a timely manner. However, there are no significant ethical concerns as we are conducting quantitative research with a public dataset.

The data set in question is an imported dataset that encompasses variables from California houses in 1990. It's relatively old and could have implications for the relevance of the findings. The methods presented throughout the study, however, can be applied to more recent data accordingly. This housing dataset was collected from the StatLib repository, a well-known and reliable source for pre-processed datasets.

Utilizing machine learning techniques in predicting house prices offers a range of advantages over traditional assessment methods. Machine learning models can significantly reduce market analysis and prediction time. These models can quickly process expansive amounts of data and identify particular patterns in real-time, enabling investors to make informed decisions in less time[5]. Moreover, machine learning can focus on big data and allow access to a wealth of information that traditional methods may fail to assess. Market datasets incorporate historical market data and various variables, including economic indicators and demographic information, providing a more comprehensive perspective of the current real estate market. Machine learning techniques also excel in accuracy and specificity, minimizing the risk of human error often present in traditional methods[5]. Applying machine learning techniques in predicting house prices has transformed the real estate industry. Traditionally, property valuation relied on simplistic models and assessments. However, machine learning has introduced a highly accurate data-driven method for forecasting certain property prices.

Literature Review

Over the past few decades, considerable effort has been devoted to leveraging both traditional and machine learning (ML) methods in real estate forecasting. The inception of machine learning in the 1990s marked a pivotal moment in the field's evolution as it began to garner recognition and underwent significant development. Various models have emerged and been explored extensively, encompassing both linear and non-linear approaches[6]. In tandem with this exploration, researchers have extended these models across diverse real estate market sectors, including commercial properties and office building rentals[7,8]. Specific machine learning models have also garnered attention in real estate forecasting. For instance, the Autoregressive Integrated Moving Average (ARIMA) is a widely recognized model for analyzing real estate dynamics[9,10]. Additionally, efforts have been made to incorporate Building Information Modeling (BIM) processes into economic assessments, particularly in the realm of real estate economics. The literature highlights a rich landscape of methodologies and techniques employed in real estate forecasting, reflecting a concerted effort to enhance predictive accuracy and inform decision-making processes within the industry.

According to (Baldominos et al, 2018)[5], the application of machine learning has the potential to identify real estate investment prospects. This innovative technology has the ability to transform how we discover and analyze opportunities in the real estate sector. The study emphasizes utilizing machine learning algorithms to analyze datasets, uncovering patterns and trends that may not be readily apparent through conventional analysis methods. By adopting this approach, not only can we expedite the identification of promising investment possibilities, but we can also leverage predictive analysis to improve decision-making in investments significantly. Similarly, (Ho et al, 2020)[11] examine how ML algorithms can predict property prices. Their study, published in the Journal of Property Research, analyzes ML models in estimating real estate values. It sheds light on the advantages and limitations of each model, making it valuable for developers, investors, and policymakers who depend on property valuations for purposes such as investment analysis and taxation. In their article published in (The Journal of Portfolio Management)[12] discuss the shift from appraisal methods to automated valuation models (AVMs). They argue that the appraisal process can become more efficient and less subjective with the emergence of "data" and advanced machine learning techniques in the real estate industry. The authors emphasize the significance of incorporating data sources, such as environmental information, into AVMs to improve their accuracy and reliability. In another study published in the Journal of Property Research, (Pérez Rave et al, 2019)[13] delve deeper into using machine learning for regression analysis of real estate prices. They focus on predictive purposes, showcasing how machine learning can provide insights into the complex relationships between various factors and real estate prices. Their research aims to benefit investors, policymakers, and urban planners. Lastly, (Zhang et al, 2017)[14] conducted a study published in Sustainability exploring factors influencing Airbnb listing prices using a weighted approach. By applying machine learning techniques to analyze data, they highlight the importance of location elements in determining property values within short-term rental markets. This study enhances our comprehension of the changing dynamics of real estate prices and the localized

factors that impact them.

In summary, the literature examined demonstrates the transformative potential of machine learning in the real estate industry. Experts and professionals can uncover valuable insights regarding market trends, property values, and investment opportunities using datasets and advanced algorithms. These advancements enhance the effectiveness and precision of real estate analysis and foster innovation in property management and investment strategies.

Methodology

This experimental research employs various machine learning models, including linear regression, support vector regression (SVR), and deep neural networks (DNNs), for a comprehensive modeling strategy. Linear regression is valued for its simplicity and interpretability, especially with linear variable relationships. SVR is robust for handling non-linear relationships, less sensitive to outliers, and can model complex relationships using kernel functions. The combination of linear regression and SVR allows for thorough analysis, determining if a simple linear model or a sophisticated method like SVR is needed for accurate predictions. DNNs effectively capture non-linear relationships and extract features from high-dimensional data, suitable for tasks like image recognition and speech understanding. However, DNNs require large datasets, are computationally expensive, prone to overfitting, and have limited interpretability. These factors were considered when selecting DNNs for real estate data analysis.

In summary, the research chose linear regression for simplicity, and SVR and DNN for robustness in complex scenarios, enabling comprehensive analysis. The study aims to find a function that accurately approximates the relationship between input features and output values, using the California Housing Prices dataset from the StatLib repository.

Linear Regression

Linear regression is pivotal in statistical modeling and is a foundational concept for various machine learning models. The model is a part of some of the most important supervised learning algorithms and proves useful when predicting quantitative data. LR is derived from Regression Analysis, a statistical analysis method, as early as 1805. Regression analysis allows for examining a relationship between a single dependent variable and one or more independent variables. In particular, linear regression remains widely used and relevant as a statistical learning tool. Furthermore, it serves as a reliable starting point for newer processes, which may be extensions of linear regression.

The mathematics of a linear regression model is rooted in the relationship it establishes between the dependent variable (often denoted as Y) and one or more independent variables (denoted as $X_1, X_2, X_3, \ldots, X_n$ for multiple regression). The basic form of a linear regression model can be expressed as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_n X_n + \varepsilon$$

Here's a breakdown of the components:

- Y: This is the dependent variable or the variable that is trying to be explained.
- $X_1, X_2, X_3, \ldots, X_n$ are the independent variables or predictors. These are the variables that could be influencing or explaining the dependent variable.
- β_0: This is the Y intercept. It represents the value of Y when all the independent variables X are equal to zero. It's where the linear line intersects with the Y axis.
- $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$: These are the coefficients of the independent variables. Each coefficient represents the change in the dependent variable for a one-unit change in the respective independent variable, assuming all other variables are held constant. These coefficients are central to interpreting the regression model.
- ε: This term represents the error or residual. It accounts for the difference between the dependent variable's actual value and the model's predicted value. The true relationship is probably not linear; other variables may cause variation in Y, and there may be measurement error. In an ideal model, ε should be as close to zero as possible, but in practice, there is always some error independent of X.

Linear regression analysis aims to estimate the values of β that best fit the given data. This is typically done through a method known as Ordinary Least Squares (OLS), which minimizes the sum of the squares of the residuals (the differences between the observed values and the values predicted by the model).

Support Vector Regression

Support Vector Regression (SVR), a type of Support Vector Machine (SVM) for regression tasks, handles both linear and non-linear relationships. It transforms data into a higher-dimensional space, enabling linear regression in complex scenarios. SVR is less sensitive to outliers and can model intricate relationships using kernel functions and the epsilon-insensitive loss function. SVR fits a line or curve within a specified threshold, epsilonε, allowing some training data errors and enhancing outlier handling. SVR uses three types of kernels to transform input data into a space where linear regression can be performed more effectively, influencing the shape of the regression line or curve.

Linear Kernel: The linear kernel is the simplest form used when the data is linearly separable. It is given by the dot product of two feature vectors. The functional form is:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{X}_i \cdot \mathbf{X}_j$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are feature vectors in the input space.

Radial Basis Function (RBF) Kernel: The RBF kernel, or the Gaussian kernel, is a popular choice for non-linear data. It is given by the exponential function of the negative squared Euclidean distance between two feature vectors, scaled by a parameter:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(-\gamma \|\mathbf{X}_i - \mathbf{X}_j\|^2\right)$$

where γ is a free parameter that determines the spread of the Gaussian distribution; higher values of γ lead to narrower peaks.

Polynomial Kernel: The polynomial kernel allows the learning of non-linear models by applying polynomial transformations to the dot product of feature vectors. The functional form is:

$$K(\mathbf{X}_i, \mathbf{X}_j) = (\alpha \mathbf{X}_i \cdot \mathbf{X}_j + c)^d$$

where α is the scale factor, c is the constant term, and d is the polynomial degree. This kernel allows the model to learn more complex relationships by considering higher-dimensional interactions between the features.

Each of these kernels has its own advantages and is suited to different types of data and regression problems. The choice of kernel depends on the specific characteristics of the data and the desired model complexity.

Deep Neural Network

In the realm of machine learning regression, a deep neural network is a model that aims to predict outcomes by understanding intricate relationships within the data. Think of it as a network of interconnected nodes or "neurons " arranged in layers. Each layer specializes in learning aspects of the data. At the core of this network, there are three types of layers:

- **Input Layer** - This is where our journey commences. Every neuron in this layer represents a feature or characteristic in our data, such as a house's size or a day's temperature.
- **Hidden Layers** - These layers reside between the input and output layers. They are referred to as "hidden" because their values are not directly observable in the data but are derived from the input. A "deep" neural network consists of layers progressively transforming the data more complexly. It's akin to passing our data through filters that extract detailed information. The greater the number of layers (and neurons within those layers), the more patterns our network can comprehend. However, finding the right balance is crucial. If there are layers, it can make the network difficult to train and more susceptible to overfitting. Overfitting occurs when the network learns the noise in the training data rather than understanding the patterns. **Output Layer** - This layer serves as the destination. When dealing with regression tasks typically, a single neuron is used in the output layer to provide a value, such as the price of a house or the amount of rainfall.

Learning involves adjusting the connections (or "weights") between neurons based on the error between predicted and actual outcomes. This adjustment uses backpropagation, where the error is passed back through the network, layer by layer, modifying the weights to minimize the error. Deep neural networks are powerful because they capture complex relationships that simpler models may miss. However, training them requires significant data and computational resources, and their inner workings can be difficult to understand, resembling a black box.

Empirical Study

Machine learning data pre-process involves several key steps, from data preparation to training and evaluating the model. Here are the steps involved in this research:

- **Data Collection:** gathering the data that will be used for model training, including identifying the independent variables (features) and the dependent variable (target).
- **Data Preprocessing:** Once the data is collected, it typically needs to be cleaned and prepared for analysis. This step includes handling missing values, removing outliers, and possibly transforming variables to fit a linear model better. In this project, we build a library of transformation functions for data handling and processing.
- **Splitting the Dataset:** The dataset is split into training and test sets. The training set is used to train the model, and the test set is used to evaluate its performance. We are using a 20% split for the test set.
- **Model Training and Evaluation:** This step trains models using preprocessed data. The training process involves finding the coefficients that minimize the error between the predicted and actual values of the dependent variable. After training, the model's performance is evaluated using the test set. Common metrics for evaluation include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). These metrics give insights into how well the model predicts and where improvements might be needed.

In the end, we can compare the performance of different models to find the best-fitting strategy for the given dataset. We will then use that data to draw conclusions and make predictions.

Data Collection

Predicting house prices is crucial in the real estate market. This paper explores the California Housing Prices dataset from the StatLib repository*. The raw data was collected from the 1990 Census, with participation from 1425.5 individuals living in various geographic locations. After excluding all block groups reporting zero entries for either independent or dependent variables, the final dataset consisted of 20,640 observations with nine distinct variables, as detailed in Table 1. The housing dataset comprises eight continuous numerical attributes and one categorical attribute, 'ocean_proximity.' The 'ocean_proximity' attribute contains five categories: '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', and 'NEAR OCEAN', indicating the relative location of the house to the ocean.

Data Preprocessing

Once the data is collected, it must be cleaned and prepared for analysis. This step includes handling missing values, removing outliers, and possibly transforming variables to fit a linear model better. In this project, we build a library of transformation functions for data handling and processing.

Feature Statistics:

Feature statistics are one of the most used statistical concepts when exploring data analysis. It touches on the numerical characteristics and properties of features within a dataset. Below, we explore the distribution, variation, and relationships between features, which assists with identifying underlying patterns, outliers, and determining the proper techniques for when data modeling occurs. We examine feature statistics to better perform data preprocessing, feature engineering, and model selection throughout the entirety of the research process.

Table 2 presents a summary of each numerical attribute. From exploring feature statistics, averages, and standard deviations can be obtained as well. We observe that the count of total bedrooms is 20433.00, while the other attributes have a count of 20640.00. This discrepancy indicates that there are missing values in the total_bedrooms feature.

The average value of median house value is 206855, and the standard deviation is 115,395.62. The std row provides the standard deviation (std), which measures the dispersion of the numerical data relative to the mean. How low or small it is - specifies how closely or largely spread out the data is around the mean. The highest price of housing is 500,001.00, and the lowest is 14,999.00. The 25%, 50%, and 75% rows display the corresponding percentage, which indicates the value below which a given percentage of observations in a group of observations fall. So, this percentage helps us understand the distribution of the data, and identify potential outliers or unusual patterns.

Visualization of Distribution / Outliers

To understand the numerical features, we plotted a histogram for each attribute, as shown in Figure 1. A histogram displays frequency distributions, showing how often each value occurs within predefined categories. It helps understand the data's shape, tendencies, variability, and potential outliers, which are useful in data exploration and machine learning analysis. After creating a histogram, we can adjust various aspects by changing its property values, useful for modifying bin properties or the overall display. From the histograms, we discovered important information.

- The housing_median_age and median_housing_value were capped.
- Many histograms are tail-heavy, meaning they extend much farther to the right of the median than to the left. Including our dependent variable, median_house_price, the plot clearly shows a left-skewed distribution. The highly skewed data can significantly impact prediction performance.
- All attributes have very different skews.
- The median_income attribute does not appear to be expressed in US dollars (USD). After gathering more information, we realized that the data has been scaled and capped at 15.00 as the highest income and the lowest income being 0.50. These numbers roughly represent tens of thousands of dollars. For example, the 4 actually means about $40,000.

Correlation

The correlation coefficient ranges from -1 to 1, indicating the strength and direction of linear relationships between variables. A coefficient close to 1 shows a strong positive correlation, but it only measures linear relationships and may miss nonlinear connections, requiring more robust tools for complex analysis.

The correlation matrix heatmap in Figure 2, generated using Python's seaborn library, reveals several variables with high correlations. Notably, the correlation between households and total_bedrooms is 0.98, and between total_bedrooms and total_rooms is 0.93. High correlations between features can cause issues like multicollinearity, where highly correlated independent variables exist in a regression model.

Figure 3 shows the correlation between the features and target variable. Correlation table 3 illustrates the relationships between features and target variables regarding housing prices. Notably,

* Please see https://www.dcc.fc.up.pt/∼ltorgo/Regression/cal$_h$*ousing.html*

No.	Features	Description
1	longitude	Longitude coordinates
2	latitude	Latitude coordinates
3	housing_median_age	Median age of homeowners within the California area
4	total_rooms	Total number of standard rooms within the California area
5	total_bedrooms	Total number of standard bedrooms within the California area
6	population	Number of people who live within the California area
7	households	Number of housing units within the California area
8	median_income	Median income of homeowners within the California area
9	ocean_proximity	Classification of varying proximities in regard to the Pacific Ocean
10	rooms_per_household	Number of rooms within each California household
11	bedrooms_per_room	Number of bedrooms within each California household
12	population_per_household	Number of people who live within each California household
13	income_cat	Income category attribute with given categories labeled from 1 to 5

Table 1 Data description of California house price dataset

Table 2 Summary of numerical attributes

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.00	20640.00	20640.00	20640.00	20433.00	20640.00	20640.00	20640.00	20640.00
mean	-119.57	35.63	28.64	2635.76	537.87	1425.48	499.54	3.87	206855.82
std	2.00	2.13	12.59	2181.62	421.38	1132.46	382.33	1.90	115395.62
min	-124.35	32.54	1.00	2.00	1.00	3.00	1.00	0.50	14999.00
25%	-121.80	33.93	18.00	1447.75	296.00	787.00	280.00	2.56	119600.00
50%	-118.49	34.26	29.00	2127.00	435.00	1166.00	409.00	3.53	179700.00
75%	-118.01	37.71	37.00	3148.00	547.00	1725.00	605.00	4.74	264725.00
max	-114.31	41.95	52.00	39320.00	6445.00	35682.00	6082.00	15.00	500001.00

median income shows the highest correlation with house price. This suggests that the median income contains the most relative information for predicting house prices in a linear model. In the upcoming section, we will develop more features to improve the predictive performance of our model based on the insight from this correlation analysis.

Table 3 Correlation between features and target variables

Numerical attribute	Correlation Coefficient
median_house_value	1.000000
median_income	0.688075
total_rooms	0.134153
housing_median_age	0.105623
households	0.065843
total_bedrooms	0.049686
population	-0.024650
longitude	-0.045967
latitude	-0.144160

Feature Engineering

We have developed three new features derived from the original variables addressing the issue of limited usefulness in their original form. For instance, the total number of rooms in a district lacks the context without corresponding information on the number of households. To overcome this limitation, we have created the following formula to generate new features.

$$\text{rooms\_per\_household} = \frac{\text{total\_rooms}}{\text{households}}$$

$$\text{bedrooms\_per\_room} = \frac{\text{total\_bedrooms}}{\text{total\_rooms}}$$

$$\text{population\_per\_household} = \frac{\text{population}}{\text{households}}$$

Additionally, we have introduced an income category attribute with 5 different categories. Category 1 ranges from 0-1.5, Category 2 ranges from 1.5-3, and so forth. In table 4, we present the correlation coefficient with the target variable, after including new features. Our result indicates that these new features have higher correlations with the target house price in comparison to the original features. For example, the correlation of rooms_per_household is 0.151948, which is higher than either the total_rooms with the correlation of 0.134153 or households with the correlation of 0.065843.

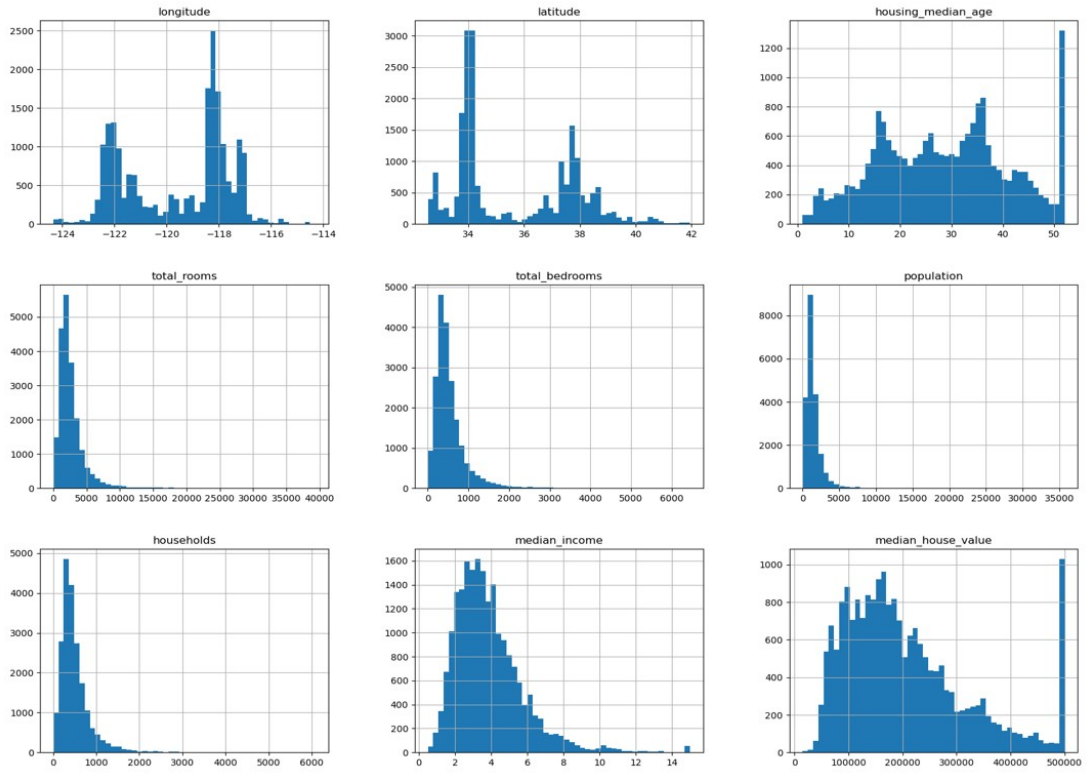

Fig. 1 Histogram of numerical attribute distributions

Table 4 Correlation between features and target variables include new variables

Numerical Attribute	Correlation Coefficient
median_house_value	1.000000
median_income	0.688075
income_cat	0.643892
rooms_per_household	0.151948
total_rooms	0.134153
housing_median_age	0.105623
households	0.065843
total_bedrooms	0.049686
population_per_household	-0.023737
population	-0.024650
longitude	-0.045967
latitude	-0.144160
bedrooms_per_room	-0.255880

Our model is a deep neural network that automatically extracts and combines features through multiple layers of nonlinear transformations. Each layer performs feature extraction, building higher-level features from lower-level ones, optimizing correlations without manual intervention. DNNs require large datasets for training, which helps the network learn robust feature representations and reduces dependence on feature selection. Large datasets also help mitigate feature correlation issues, allowing the network to find globally optimal feature combinations. We use a layer-wise decreasing structure, where the number of neurons in each layer gradually decreases, forming a pyramid-shaped network. This reduces feature dimensions in deeper layers, compressing features by extracting information from many raw features initially and gradually reducing the feature count, retaining the most useful representations.

Geographical information

To visualize the geographical data, we created a scatter plot of all districts, as shown in Figure 4. In this plot, we have used darker colors to highlight high-density areas. The radius of each circle corresponds to its respective district population. The color indicates the house price, with red representing more expensive areas and blue representing more affordable ones. Larger circles indicate areas with high populations. This plot may indicate a strong relationship between housing prices and location factors, such as proximity to the ocean as well as population density.

Handling Text and Categorical Attributes

Most machine learning algorithms prefer numerical inputs.

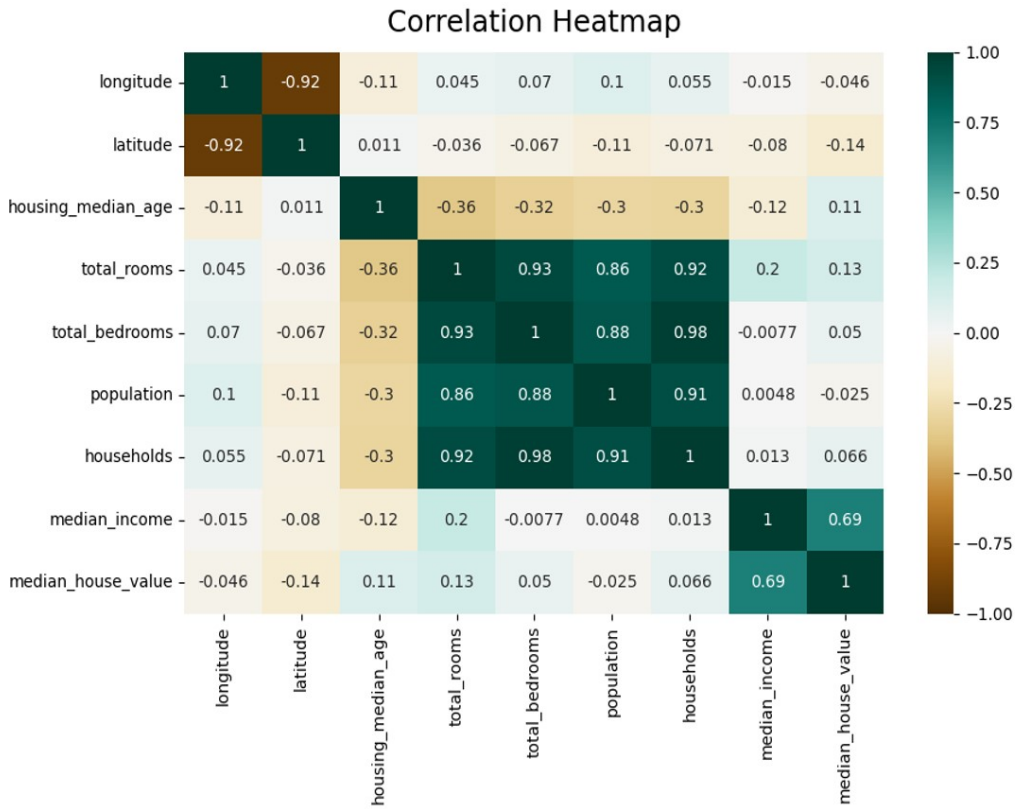

Fig. 2 Correlation Heatmap of numerical attributes

Therefore, converting text attributes into numerical format is essential. Our dataset's 'ocean_proximity' attribute contains categorical values: ['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND']. To convert these, we use encoding techniques like ordinal encoding and one-hot encoding.

Ordinal encoding assigns each category a specific integer but assumes an ordered relationship, which may introduce unintended patterns. To avoid this, we use one-hot encoding, creating a binary attribute for each category. Each observation is represented by a sparse matrix where only one attribute is 1 (hot) and the others are 0 (cold), ensuring no inherent order among categories.

Scikit-learn's OneHotEncoder function transforms categorical values into a sparse matrix, with each row corresponding to an observation and each column representing a category. After one-hot encoding, we obtain a matrix with 5 columns, filled mostly with 0s except for a single 1 per row indicating the observation's category. This ensures categorical attributes are appropriately represented numerically without introducing biases.

Missing values

Most machine learning algorithms can't handle missing values. In our dataset, both original and created attributes contain missing values. Researchers typically address this by either deleting or imputing the missing values. Deleting involves removing observations or attributes with missing values, which can result in data loss. Imputation replaces missing values with substitutes like the mean, median, or mode, retaining observations and attributes.

In our dataset, only 1% of the values are missing in the 'total_bedrooms' attribute. We chose imputation, replacing missing values with the median. Using the median ensures the imputed values represent the data distribution and mitigates outliers' impact. This approach maintains data completeness and integrity, facilitating robust model training and analysis.

Feature Scaling

One crucial transformation in machine learning projects is feature scaling, as algorithms underperform with input attributes of different scales. This is true for our house price dataset, where attributes like median income (0 to 15) vary significantly from

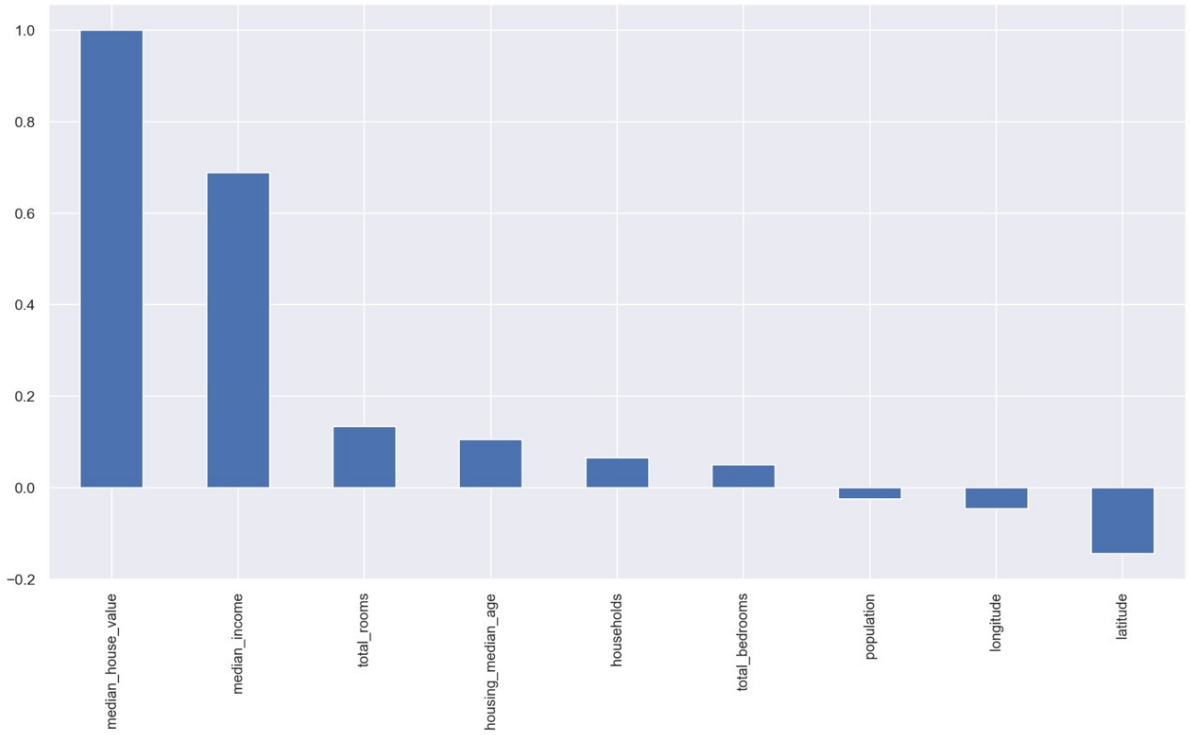

Fig. 3 Correlation between features and target

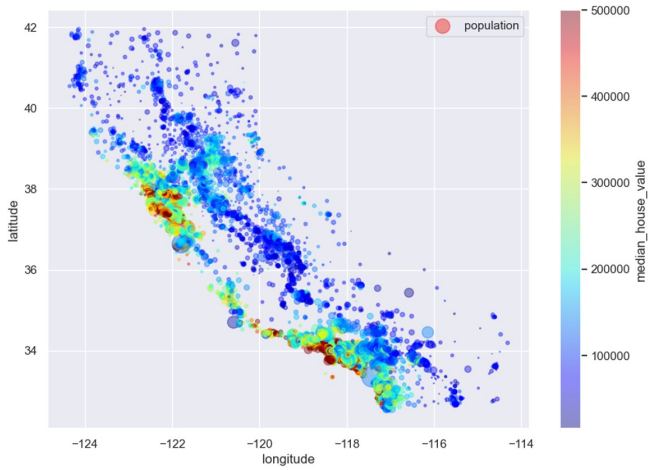

Fig. 4 Scatter plot of California districts with population

household population (0 to 6,000).

Two common methods to standardize scales are normalization and standardization. Normalization, or min-max scaling, rescales each element to fall within 0 to 1 by subtracting the minimum value and dividing by the range (maximum minus minimum). The normalization formula is:

$$x_{\text{scaled}} = \frac{x - \min}{\max - \min}$$

Alternatively, standardization involves rescaling the features to have a mean of 0 and a standard deviation of 1. This is accomplished by subtracting the mean and then dividing by the standard deviation. The formula for standardization is shown as follows:

$$x_{\text{scaled}} = \frac{x - \text{mean}}{\text{standard deviation}}$$

In our study, we use Scikit-learn's `StandardScaler` to perform feature scaling [15]. This library standardizes features to the same scale. We created a function called `scaler`, using `StandardScaler` to fit and transform our numerical attributes. This ensures our model is not biased towards larger-scale features, enhancing performance and stability.

Splitting the Dataset

After training a model, it's essential to assess its ability to generalize by splitting the dataset into training and test sets. The model is trained on the training set and evaluated on the test set. The Out-of-Sample error measures performance on new data, while the In-Sample error measures performance on training

data. A low In-Sample error but a high Out-of-Sample error suggests overfitting.

Traditionally, random shuffling splits the dataset, but it can lead to class imbalances. Stratified shuffling addresses this by ensuring proportional class representation in both sets. For this study, we used the StratifiedShuffleSplit() method based on the 'income_cat' variable, providing representative samples for training and evaluation, enhancing model robustness and reliability.

Performance Metrics

Performance metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are critical in evaluating the accuracy and reliability of predictive models. These metrics measure the difference between the predicted values by the model and the actual values in the dataset, thus providing insights into the model's performance.

Mean Squared Error (MSE) is calculated as the average of the squared differences between the predicted and actual values, with the formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Here, n is the number of observations, y_i represents the actual values, and $\hat{y}_i$ denotes the predicted values. MSE is sensitive to outliers because the squaring of the errors exaggerates larger discrepancies.

Mean Absolute Error (MAE) measures the average of the absolute differences between predicted and actual values, without squaring, as shown in:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

This makes MAE more robust against outliers compared to MSE, as it treats all errors on the same linear scale.

Root Mean Squared Error (RMSE), the square root of MSE, is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE is particularly useful because it scales the errors back to the original units of the data, making it more interpretable than MSE. Like MSE, RMSE is sensitive to outliers due to the squaring of errors.

These metrics are vital for model evaluation and selection, as they provide a clear measure of how closely a model's predictions match the actual data. Lower values of MSE, MAE, and RMSE indicate better model performance. By quantifying prediction accuracy, these metrics facilitate model optimization and guide decision-making in applications such as real estate analytics, where accurate price forecasting and market analysis are crucial. In this research, we will mainly use RMSE to compare the predictive power of different models.

Model Training and Results

Linear Regression Model

In our study, we use a linear regression model from the Scikit-learn library (Pedregosa et al., 2011). The model is created with the linear regression class and trained using the fit() method, which establishes a linear relationship between input and target variables. After training, we generate predictions using the predict() method and assess performance with the Mean Squared Error (MSE) calculated by the mean_squared_error() method from sklearn.metrics. MSE measures the average squared differences between actual and predicted values, providing a quantitative performance measure. Our goal is to evaluate the model's ability to identify patterns and make accurate predictions regarding house prices.

Support Vector Machine

In this study, we explored support vector regression (SVR) models with various kernels using the Scikit-learn library (Pedregosa et al., 2011). We tested linear, polynomial, and radial basis function (RBF) kernels on our house price dataset. Each SVR model used different kernels and a regularization parameter (C) of 10,000. The models were trained with the training features and target values, and their predictions were evaluated on the test set.

SVR involves several hyperparameters, including kernel types, regularization parameters, Gamma (gamma), Degree (degree), and Coefficient parameters. The choice of kernel affects the decision boundary: linear kernels suit linearly separable data, RBF kernels handle non-linear data, and polynomial kernels fit polynomial decision boundaries. Regularization parameters control the trade-off between training error and decision boundary smoothness, impacting the model's performance and generalization. Selecting appropriate hyperparameters is crucial for optimizing the SVR model.

Deep Neural Network

In our research, we use the Sequential() model from TensorFlow, which stacks layers with one input tensor and one output tensor per layer. We include four hidden layers with 128, 64, 32, and 16 neurons, respectively. Figure 5 outlines our deep neural network architecture, including trainable hyperparameters, layers, and their shapes.

We use the Rectified Linear Unit (ReLU) activation function to introduce non-linearity, and the final dense layer has a single

neuron with no activation function for regression tasks. We evaluate performance with the Mean Squared Error (MSE) loss function and use the 'Adam' optimizer for training. Hyperparameters such as batch size (64), epochs (300), and the number of hidden layers are adjusted, with EarlyStopping callbacks monitoring validation loss to halt training after 5 epochs without improvement.

Figure 6 shows the training history, with loss values (MSE) for training and test sets across epochs. Analyzing loss helps determine if the model is converging. Decreasing and stabilizing loss indicates effective learning, while increasing loss might signal overfitting or underfitting. A significant training-test loss disparity suggests overfitting, while low training loss with high test loss suggests underfitting.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               2304

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 16)                528

 dense_4 (Dense)             (None, 1)                 17

=================================================================
Total params: 13185 (51.50 KB)
Trainable params: 13185 (51.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. 5 Overview of our deep neural network architecture

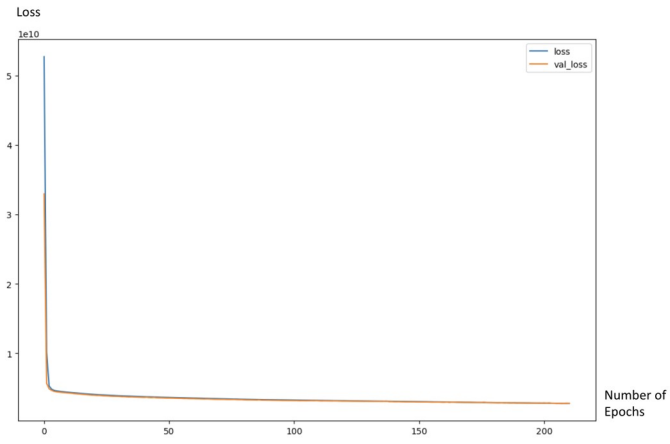

Fig. 6 loss during the training of a neural network

Discussion

In this paper, the empirical result successfully reviewed and evaluated the performance of different regression models, including linear regression, support vector machine with linear, polynomial, radial basis function kernels, and deep neural network. The RMSE was used as a metric to assess the accuracy of these models in predicting house price values. In the training set, the linear regression models demonstrate an RMSE of 67795.88. The SVR with linear, polynomial, and RBF kernels shows RMSE values of 69470.10, 63527.317, and 58182.378, respectively. Our deep neural network model shows an RMSE of 59912.37 in the training set. Upon evaluating models in the test set, the result continued to provide insight into the generalization performance. The linear regression models demonstrate an RMSE of 66638.348. The SVR with linear, polynomial, and RBF kernels shows RMSE values of 68070.48, 66299.077, and 57404.80, respectively, and our DNN model shows an RMSE of 59262.93 in the test set.

Results above show that the SVR model with RBF and DNN demonstrates comparatively lower RMSE values on both the training and test sets when contrasted with other models, suggesting their superior accuracy. Conversely, the linear regression model exhibits a relatively higher RMSE, implying its predictions deviate further from actual values. This discrepancy may stem from the linear model's inability to capture the underlying intricacies within our house price dataset fully. Moreover, the SVR with a linear kernel yields a slightly elevated RMSE compared to both linear regression and alternative SVR kernels, hinting at its failure to capture complex relationships adequately. Conversely, the polynomial kernel outperforms the linear kernel, showcasing its efficacy in capturing nuanced relationships within our dataset. Notably, the SVR with RBF kernels attains the lowest RMSE in the training set, underscoring its adaptability in modeling nonlinear relationships. Although the DNN model's RMSE slightly exceeds that of the SVR with RBF, it still demonstrates strong predictive capabilities, particularly in capturing nonlinear patterns.

The choice of kernel significantly impacts the SVR performance, with the RBF kernel emerging as the most effective for regression tasks. Its outstanding performance makes it a robust candidate for future real-world applications in house price predictions. The linear and polynomial SVR kernels are relatively more competitive than linear regression but have lower RMSE than RBF kernels, which indicates the limitations of these two kernels in capturing the underlying patterns in the dataset. The DNN model performs well but may benefit from additional optimization, architecture changes, or training hyper-parameters. In conclusion, the choice of model and its hyper-parameters significantly impacts the model's predictive power. Our empirical study shows that SVR with an RBF kernel is the most well-suited model for this regression task. The findings from our empirical study provide a foundation for future work, including model optimization and feature engineering to improve model performance, and further investigation into the characteristics of the dataset or alternative model architectures might also be helpful.

In terms of limitations throughout the study, there are a few. The first is the dataset's age. The imported dataset used in this study encompasses variables from California houses in 1990. Since it's relatively old, it could have implications for the relevance of the findings. The methods presented throughout the study, however, can be applied to more recent data accordingly. Another limitation is the strong randomness in Deep Neural Network. Deep neural networks exhibit inherent randomness, which may lead to variations in results. Lastly, a notable limitation is the lack of interpretability associated with some machine learning models, particularly deep neural networks. It can be challenging for humans to comprehend each model's predictions and interpret how they came to these conclusions.

Overall, this research demonstrates for real estate investors and local real estate agents, that it will be beneficial to incorporate SVR with RBF kernel when predicting house values and market trends. It will also assist Short Term Rental / Airbnb investors in CAP (Capitalization Rate) analysis, risk assessments and demand forecasting.

References

1 G. Georgiev, B. Gupta and T. Kunkel, *Journal of Portfolio Management*, 2003, **29**, 28–34.

2 A. W. Lo, *Hedge Funds, Systemic Risk, and the Financial Crisis of 2007-2008: Written Testimony for the House Oversight Committee Hearing on Hedge Funds*, Research Methods & Methodology in Accounting eJournal, 2008.

3 L. Choy and W. Ho, *Land*, 2023, **12**, 740.

4 J. Kahr and M. C. Thomsett, *Real estate market valuation and analysis*, John Wiley & Sons, 2006.

5 A. Baldominos, I. Blanco, A. J. Moreno, R. Iturrarte, Bernárdez and C. Afonso, *Applied Sciences*, 2018, **8**, 2321.

6 R. Füss, M. Stein and J. Zietz, *Journal of Real Estate Finance and Economics*, 2012, **45**, 653–674.

7 K. H. D. Ho, S. Rengarajan and J. L. Glascock, *Journal of Property Investment & Finance*, 2014, **32**, 384–403.

8 M. Tonelli, S. Cowley and T. Boyd, *Journal of Real Estate Research*, 2004, **26**, 77–98.

9 S. Stevenson, *Journal of Real Estate Research*, 2007, **29**, 95–111.

10 R. Martin and T. Setzer, *Journal of Property Investment & Finance*, 2014, **32**, 610–629.

11 W. K. Ho *et al.*, *Journal of Property Research*, 2020, **38**, 1–23.

12 N. Kok *et al.*, *The Journal of Portfolio Management*, 2017, **43**, 202–211.

13 J. I. Pérez-Rave *et al.*, *Journal of Property Research*, 2019, **36**, 59–96.

14 Z. Zhang *et al.*, *Sustainability*, 2017, **9**, 1635.

15 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg and J. Vanderplas, *The Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

Appendix

https://github.com/DancingByte/Deep-Learning-Real-Estate-Forecast/blob/main/Paper$_{A}ppendix.pdf$.