CS5296 Project
House Price Prediction Using PySpark

Name	SID
LAN JINGSEN	58158499
MU KEXIN	57992056
WEI ZIBO	57978062

Abstract

The housing market is a critical sector for economic stability and individual wealth, making accurate house price prediction a valuable endeavor. Our project, titled "House Price Prediction Using PySpark," leverages machine learning techniques within the Apache Spark framework to forecast housing prices. Utilizing a comprehensive dataset from the California Housing dataset, we preprocessed the data to handle missing values and engineered features to enhance model performance. We employed various regression models, including Linear Regression and Random Forest Regression, to identify the most predictive attributes. Additionally, we explored a Fully Connected Neural Network architecture using PyTorch for deep learning-based predictions. Our findings indicate that ensemble methods like Random Forest and deep learning models provide superior accuracy, with the Random Forest achieving the best RMSE of 0.5620 and the neural network offering a nuanced understanding of complex patterns. This project not only contributes to the field of real estate analytics but also presents a robust and scalable solution for stakeholders seeking to predict housing prices with precision.

Index Terms—PySpark, machine learning, deep learning, real estate analytics.

I. INTRODUCTION

The subject of our project is the prediction of housing prices, a challenge that is both economically significant and analytically complex. The context of this project is set against the backdrop of a dynamic real estate market where price forecasting is essential for buyers, sellers, investors, and policymakers. The motivation for this work is to provide a data-driven approach to price prediction that can be scaled and applied in real-world scenarios.

Our methodology involves the use of PySpark, a powerful tool for big data processing, to handle the large and complex dataset efficiently. We have set up a machine learning pipeline that includes data preprocessing, feature engineering, model training, and evaluation. The importance of the problem is addressed through the use of both traditional machine learning algorithms and deep learning techniques, which are compared for their effectiveness in predicting housing prices.

The major findings of our project include the identification of key features that influence housing prices and the development of models that can predict these prices with high accuracy. Specifically, we found that models leveraging ensemble learning techniques and deep learning architectures outperformed other methods in terms of predictive accuracy, as measured by RMSE, MAE, and R2 metrics.

Our study presents a clear and logical narrative that begins with an exploration of the dataset and proceeds through a systematic approach to model development and evaluation. The project is interesting not only because of the economic implications of the findings but also because of the technical depth and innovation in the methods employed.

II.EXPERIMENTS

A. Dataset Description and Preprocessing

1) Dataset Overview:

The California Housing dataset, used in this study, contains various features such as median age, total rooms, and housing prices among others. The initial step in our analysis involved understanding the dataset's composition and structure.

The California Housing Dataset is a commonly used public dataset widely used in research and practice in the fields of machine learning and data science. It provides information about homes in different regions of California and is designed to help researchers and practitioners explore and resolve issues related to home prices and characteristics.

This dataset was originally based on data from the 1990 US Census, collected and compiled by Sigal Saharani and Marcos de Pedro et al. It encompasses a range of characteristics that cover the properties of the house and surrounding environmental factors in each area. These characteristics include population density, average income, age of homes, average number of rooms, average number of bedrooms, location, and more.

In addition to features, the dataset also provides the median house price in each region as the target variable. This allows researchers to use this dataset to train and evaluate regression models. The dataset description can be found in the Table 1.

Table 1. The dataset description.

feature name	feature description
Housing Median Age	It is the median age of the people that belong to a block group. Note that the median is the value that lies at the midpoint of a frequency distribution of observed values
Housing Median Age	It is the median age of the people that belong to a block group. Note that the median is the value that lies at the midpoint of a frequency distribution of observed values
Total Rooms	It is the total number of rooms in the houses per block group
Total Bedrooms	It is the total number of bedrooms in the houses per block group
Population	It is the number of inhabitants of a block group
Median House Value	It is the dependent variable and refers to the median house value per block group
Latitude	refers to the angular distance of a geographic place east or west of the earth's equator for each block group
Longitude	refers to the angular distance of a geographic place north or south of the earth's equator for each block group

We first define the schema of the data:

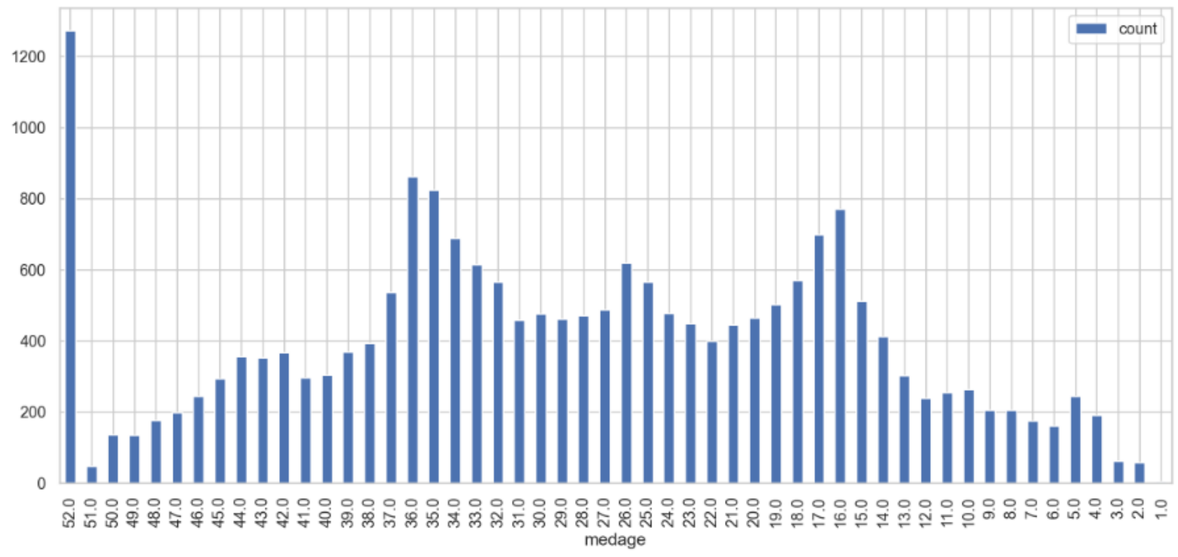
```python
# define the schema, corresponding to a line in the csv data file.
schema = StructType([
    StructField("long", FloatType(), nullable=True),
    StructField("lat", FloatType(), nullable=True),
    StructField("medage", FloatType(), nullable=True),
    StructField("totrooms", FloatType(), nullable=True),
    StructField("totbdrms", FloatType(), nullable=True),
    StructField("pop", FloatType(), nullable=True),
    StructField("houshlds", FloatType(), nullable=True),
    StructField("medinc", FloatType(), nullable=True),
    StructField("medhv", FloatType(), nullable=True)]
)
```

Then we use Spark to load the dataset.

```python
# Load housing data
housing_df = spark.read.csv(path=HOUSING_DATA, schema=schema).cache()
```

We also briefly explored the data characteristics of the dataset, such as the structure of the dataset and the distribution of the average age of houses.

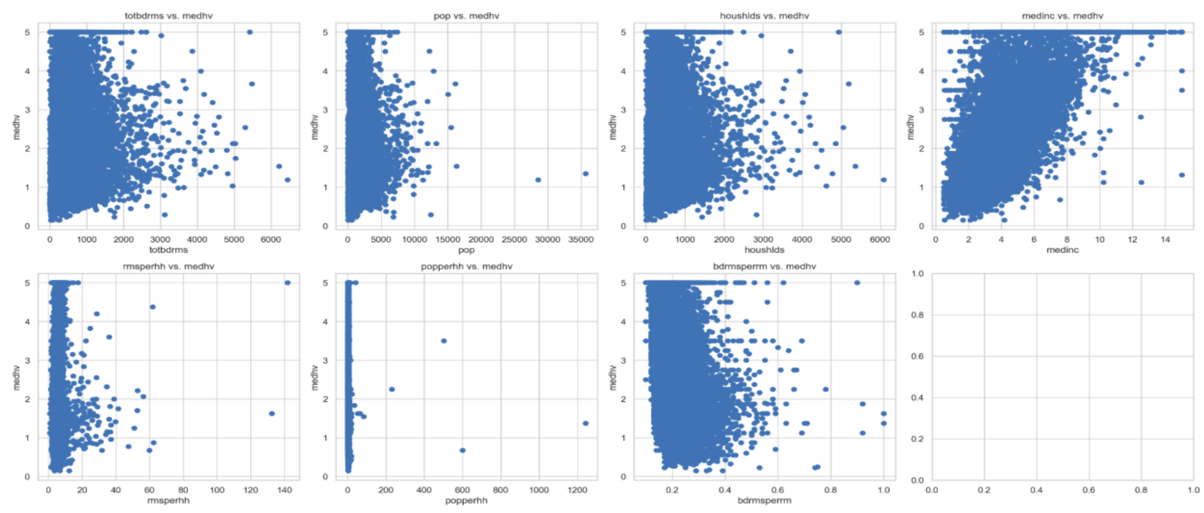
```
+-------+-------+---------+--------+---------+--------+-------+-----------+
|summary| medage| totrooms|totbdrms|      pop|houshlds| medinc|      medhv|
+-------+-------+---------+--------+---------+--------+-------+-----------+
|  count|20640.0|  20640.0| 20640.0|  20640.0| 20640.0|20640.0|    20640.0|
|   mean|28.6395|2635.7631| 537.898|1425.4767|499.5397| 3.8707|206855.8169|
| stddev|12.5856|2181.6153|421.2479|1132.4621|382.3298| 1.8998|115395.6159|
|    min|    1.0|      2.0|     1.0|      3.0|     1.0| 0.4999|    14999.0|
|    max|   52.0|  39320.0|  6445.0|  35682.0|  6082.0|15.0001|   500001.0|
+-------+-------+---------+--------+---------+--------+-------+-----------+
```



2) Data Preprocessing:

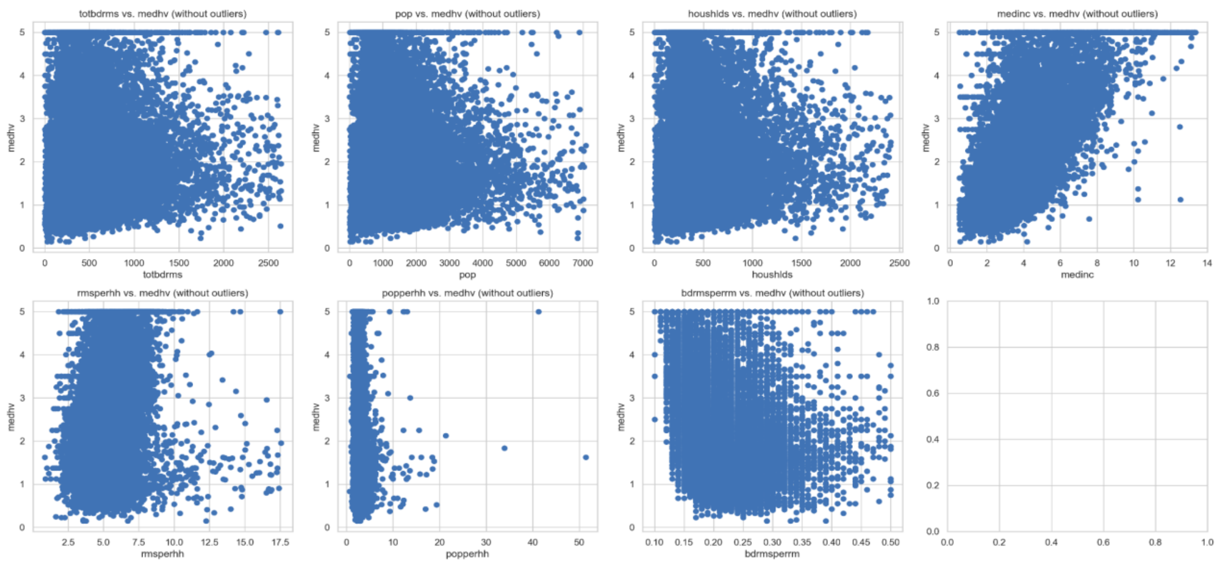
We began with data preprocessing to ensure accuracy in our models. This included handling missing values, normalizing the data, and creating new features that could be significant predictors for housing prices.

These two pictures compare the data distribution before and after cleaning.



There is some noise in the data distribution in the California Housing Dataset. Feature values in the data set may contain outliers or inaccurate data points due to possible measurement errors, data entry errors, or other factors during the data collection process.

To understand whether there is noise in the data distribution, we perform some basic statistical analysis and visual exploration to find outliers or unusual data patterns.



We finally scale the data using StandardScaler. The input columns are the features, and the output column with the rescaled that will be included in the scaled_df will be named "features_scaled".

B. Feature Engineering

1) Feature Selection:

We selected features that could potentially influence housing prices, such as the number of rooms, population, and median income.

2) Feature Transformation:

New features were engineered to better capture the essence of the data. For instance, we calculated the number of rooms per household and the population per household to understand the density and spaciousness of the housing.

We have adjusted the values in medianHouseValue, and we add the following columns to the data set:

- Rooms per household which refers to the number of rooms in households per block group.
- Population per household which basically gives us an indication of how many people live in households per block group.
- Bedrooms per room which will give us an idea about how many rooms are bedrooms per block group.

C. Model Development

In the model development stage, we used the linear regression model and random forest model in the PySpark library, and developed a fully connected neural network model using the torch library.

1) Linear Regression:

The linear regression model is a classic regression algorithm used to establish a linear relationship between features and target variables. In PySpark's MLlib library, we used a linear regression model to predict house prices. Linear regression models perform well in handling a large number of features and regression tasks. We can use the coefficients of the linear regression model to explain the extent to which different characteristics affect house prices.

2) Random Forest:

The random forest model is an ensemble learning method based on decision trees, and a corresponding implementation is also provided in the MLlib library of PySpark. We chose the random forest model as an alternative to the regression model. The random forest model makes predictions by combining multiple decision trees and can handle a large number of features and non-linear relationships. It has good generalization ability and resistance to overfitting.

3) Fully Connected Neural Network:

In addition to traditional machine learning models, we also used the torch library to develop a fully connected neural network model. Fully connected neural network is one of the commonly used model types in deep learning. Our neural network model adopts multiple fully connected layers and uses the ReLU activation function to predict house prices. By training a neural network model, it can automatically learn feature representations and complex nonlinear relationships from data.

These models have different advantages and characteristics in house price prediction tasks. Linear regression models are simple and easy to interpret, and are suitable for situations where a linear relationship is obvious. The random forest model can handle nonlinear relationships and feature interactions and performs well on complex data sets. The fully connected neural network model has strong learning ability and flexibility and can adapt to various complex data patterns.

D. Model Training and Tuning

1) Training Process:

Our models were trained on the preprocessed and feature-engineered dataset. We used Spark’s distributed computing capabilities to handle the large-scale data processing.

We use the randomSplit() method to randomly divide the training set and test set.

2) Hyperparameter Tuning:

We performed hyperparameter tuning to optimize model performance.

For the Random Forest model, we tuned the bins, maximum depth and the number of trees.

maxDepths (Maximum Depth): This parameter specifies the maximum depth for each tree in the forest. The depth of a tree determines the number of splits in a decision tree. Limiting the depth can reduce the risk of overfitting but may also increase the risk of underfitting.

numTrees (Number of Trees):This parameter determines the number of decision trees in the random forest. Increasing the number of trees can improve the model's performance because it increases the diversity of the model and the reliability of the voting. However, too many trees can increase computational costs and lead to slow convergence. Typically, a balance needs to be found to ensure there are enough trees for stable results without making the computation process too time-consuming.

Bins are used to segment continuous values into multiple intervals, which can simplify the decision-making process and reduce computational effort. Smaller values mean finer granularity, which can lead to a more complex model and a higher risk of overfitting; larger values mean coarser granularity, which can lead to a simpler model but may fail to capture all the details of the data.

For Linear Regression, we focused on regParams, elasticNetParams and maxIters.

maxIters: This parameter determines the maximum number of times an algorithm can run before stopping the search for the best model. If the algorithm does not find the best solution within this number of iterations, it will cease.

regParams: This parameter controls the strength of regularization, which is a method to reduce the complexity of the model and prevent overfitting. The higher the value of regParams, the stronger the regularization, which tends to make the model's parameters smaller.

elasticNetParams: This parameter balances the use of L1 and L2 regularization. L1 regularization can result in sparse solutions (many parameters being zero), while L2 regularization tends to keep parameter values close to zero but usually not exactly zero. The value of elasticNetParams determines whether to use more L1 regularization (values closer to 1) or L2 regularization (values closer to 0).

E. Performance Evaluation

1) Evaluation Metrics:

We used Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2) to evaluate the performance of our models.

RMSE is one of the commonly used performance evaluation indicators in regression models. It measures the standard deviation of the average error between the model's predicted values and the true values. The smaller the value of RMSE, the smaller the prediction error of the model and the better the performance of the model.

MAE is another commonly used regression model performance evaluation metric. It calculates the mean absolute error between the model's predicted values and the true values. Unlike RMSE, MAE does not consider the square of the error and is therefore not sensitive to outliers. A smaller MAE value indicates that the model's prediction error is smaller and the model's performance is better.

R2 is an indicator used to measure the goodness of fit of the regression model. It indicates the percentage of variability in the dependent variable that is explained by the model. The value of R2 ranges from 0 to 1, where 0 means that the model cannot explain any variability in the target variable, and 1 means that the model perfectly explains the variability of the target variable. A higher R2 value indicates that the model has a stronger ability to explain the target variable and the model has better performance.

2) Observations:

Table 2 shows the evaluation results of the three models on the test set.

Table 2. The evaluation results of the three models on the test set.

models	RMSE	MAE	R^2
LinearRegression	0.6665	0.4861	0.6374
RandomForest	0.5620	0.3913	0.7435
Neutral Network	0.6574	0.4723	0.6369

Based on the evaluation metrics, we can see that the RandomForest model performs best in all three metrics. It has the lowest RMSE (root mean square error) and MAE (mean absolute error), and has the highest R^2 (coefficient of determination). This shows that the RandomForest model has the best performance in predicting the target variable. Followed by the LinearRegression model, its performance in RMSE and MAE is slightly worse than the RandomForest model, but still better than the Neutral Network model. However, in terms of R^2, the LinearRegression model performs slightly lower than the Neutral Network model. Finally, there is the Neutral Network model, which performs slightly worse than the LinearRegression model and the RandomForest model in terms of RMSE and MAE, and slightly higher than the LinearRegression model in terms of R^2.

To sum up, according to the given evaluation indicators, the RandomForest model performs best among these models, followed by the LinearRegression model and the Neutral Network model performs the worst.

F. Discussion

1) Model Interpretation:

During the training process, we found that the random forest model had the best interpretability and the smallest error. We analyzed the results based on the model principles.

Random forest is an ensemble learning method that consists of multiple decision trees. During the training process, each decision tree is obtained by random sampling (bootstrap) with replacement on the original data, and random subsets of features are also selected. When making predictions, Random Forest aggregates the predictions of each decision tree. Since each decision tree is trained independently, they are independent of each other, which makes the random forest well able to deal with the overfitting problem.

Random forests have some advantages in terms of interpretability. First, since each decision tree is trained based on a subset of features, the selection of features for each decision tree may be different, which allows us to infer which features are important for prediction by observing the consistency of multiple decision trees. The results have the greatest impact. Secondly, since random forest is an ensemble model based on decision trees, the decision tree itself has a certain interpretability. We can explain the decision-making process of prediction results along the path of the decision tree, starting from the root node and understanding how the model predicts based on different features by observing the splitting rules of each node.

To sum up, the random forest model uses multiple independent decision trees during the training process, and uses randomness and integration to improve the interpretability and generalization ability of the model.

2) Limitations and Future Work:

In our experiments, the performance of the neural network was poor, possibly limited by parameter tuning. Neural networks usually have a large number of parameters that need to be adjusted, including network structure, activation function, learning rate, etc. In the future, we may use more in-depth parameter tuning algorithms, use attention mechanisms, apply data enhancement technology, etc. to improve performance.

III. CONCLUSION

The core objective of this project is to predict housing prices, a challenge that is both economically significant and analytically complex. Against the backdrop of a dynamic real estate market, where forecasting prices is crucial for buyers, sellers, investors, and policymakers, the project aims to provide a data-driven approach to price prediction that can be scaled and applied in real-world scenarios.

We employed PySpark, a powerful tool for big data processing, to efficiently handle the large and complex dataset. By establishing a machine learning pipeline that includes data preprocessing, feature engineering, model training, and evaluation, we utilized both traditional machine learning algorithms and deep learning techniques to compare their effectiveness in predicting housing prices.

The project's key findings include the identification of key features that influence housing prices and the development of models capable of predicting these prices with high accuracy. Specifically, models that leverage ensemble learning techniques and deep learning architectures outperformed other methods in terms of predictive accuracy, as measured by RMSE, MAE, and R^2 metrics.

The experimental section began with a description and preprocessing of the California Housing dataset, which contains various features such as median age, total rooms, and housing prices. The preprocessing steps included handling missing values, data normalization, and the creation of new features that could significantly predict housing prices.

In the feature engineering phase, we selected features that could potentially influence housing prices, such as the number of rooms, population, and median income, and engineered new features to better capture the essence of the data. During the model development stage, we utilized linear regression and random forest models from the PySpark library and developed a fully connected neural network model using the torch library.

The models were trained on the preprocessed and feature-engineered dataset, leveraging Spark's distributed computing capabilities for large-scale data processing. Hyperparameter tuning was performed to optimize model performance, with adjustments made to parameters such as the maximum depth and the number of trees for the Random Forest model, and regParams and elasticNetParams for Linear Regression.

Performance evaluation was conducted using RMSE, MAE, and R^2 metrics to assess the performance of our models. The results indicated that the Random Forest model performed best across all three metrics, with the lowest RMSE and MAE and the highest R^2.

In the discussion section, it was noted that the Random Forest model not only had the best interpretability but also the smallest error. However, the performance of the neural network was poor, likely limited by parameter tuning. Future work may involve more in-depth parameter tuning algorithms, the use of attention mechanisms, and the application of data augmentation techniques to enhance performance.

In conclusion, our project not only demonstrated technical depth and innovation but also held significant economic implications. Through a systematic methodology and rigorous experimental design, we provided a tool for housing price prediction, offering valuable insights to participants in the real estate market.

ARTIFACT APPENDIX

1. Project Overview

CS5296 Cloud Computing— Theory and Practice, Spring 2024 Group 10 Project

The main code of this project is in: `Project.ipynb`

The data set storage path is: `cal_housing\CaliforniaHousing\cal_housing.data`

2. Github Repository

The link to the Github repository of the project is as follow:

`https://github.com/Ysenniko/CityU_CS5296_CloudComputing_Project`

3. Detailed Installation Steps

1) Clone the Repository: Users need to clone the project from GitHub to local. Specific command:

```
git clone https://github.com/Ysenniko/CityU_CS5296_CloudComputing_Project.git
```

2) Install Dependencies: The project requires the following dependencies to be installed:

- Recommended openjdk version: v17.0
- Recommended python version: python==3.9
- pyspark==3.5.1
- numpy
- pandas
- pillow
- seaborn
- torch
- torchvision
- torchaudio
- scikit-learn

All dependencies in the `requirements.txt` file can be installed using the following command:

```
pip install -r requirements.txt
```

Please make sure you have changed to the directory containing the requirements.txt file on the command line and then run the above command.

4. How to Run Our Project

After the code running environment is set up, enter the `Project.ipynb` file. This file contains the main code implementation of the project. The project can be reproduced and verified by running the code in this Jupyter Notebook.