Student Name: Usama Masood

Student ID: S3355014

Assignment Title: Used Car Price Prediction Using Machine Learning Models

Abstract:

The goal of this study is to find out how models based on machine learning can be used to guess how much-used cars will cost. Different models, like Linear Regression, Ridge Regression, KNeighborsRegressor, DecisionTreeRegressor, and RandomForestRegressor, were tested using cross-validation. R^2, RMSE, and MAE were used to measure performance on both the training set and the test set. All of the ranking scales showed that the RandomForestRegressor model did the best on the test set. The R^2 value was 0.8747, the RMSE value was 0.9092, and the MAE value was 0.6456. The DecisionTreeRegressor did very well too, with an R^2 of 0.7798, an RMSE of 1.2049, and an MAE of 0.8464. The KNeighborsRegressor fit the data too well, as shown by an R^2 of 0 on the training set and a much lower R^2 of 0.7022 on the test set. Ridge Regression and Linear Regression were both correct, but the ensemble methods were better. In this case, RandomForestRegressor and other ensemble methods do a great job of guessing how much a used car will cost. This is likely because they can find connections in the information that aren't simple. Next time, scientists might look into more hyperparameter optimization and other group methods that can help make better guesses.

Introduction:

Problem Statement and Significance:

Used automobile sales have skyrocketed in recent years. Reasons for this include the need for more affordable transportation options, growing environmental concerns, and shifts in the economy. All parties involved—buyers, owners, and dealers—benefit from accurate used vehicle price predictions because it allows them to make informed decisions. Yet, there are a lot of factors that may affect an automobile's worth, including its age, mileage, make, model, condition, and market demand, making it difficult to predict used car pricing. We need to examine these things using complex mathematical approaches if we want to make reliable forecasts.

Literature Review:

Using various machine learning techniques, some research attempted to predict the value of secondhand automobiles. For instance, Samruddhi and Ashok Kumar (2020) achieved a 0.85 using the K-Nearest Neighbor (KNN) approach. They discovered that a regression model had lower accuracy (0.71) when tested with the same data. They used K-Fold cross-validation to determine that KNN was the most suitable model for their dataset.

"Car Price Prediction using Machine Learning Techniques" was the title of the research paper by Enis Gegic et al., which investigated the potential applications of ANN, SVM, and Random Forests. They categorized vehicles as inexpensive, moderate, and costly. Their findings showed that SVM performed better for both low- and high-priced vehicles, but ANN performed better for vehicles in the medium-price bracket. Their model, which has a total accuracy of 0.8704, shows that using more than one machine learning method together makes classification work better. K. Noor et al. (2017) used machine learning to try to guess what car prices would be in the future. They used a Multiple

Regression Model and picked the right predictor components to make correct estimates. With an accuracy of 0.98, their model showed that feature selection works by making accurate predictions.

Technology Used:

Several powerful technologies and tools used in data science and machine learning are included in this project's Python code.

1. **Python:** Python is very simple to learn and has a user-friendly interface, I decided to make use of it for my report. Regarding data evaluation and machine learning, it offers a full set of features.
2. **Pandas:** A vital data processing technology, Pandas enables the control and agency of tabular information, consisting of DataFrames. It was utilized to clean up, manipulate data, and do experimental data analysis for this project.
3. **NumPy:** It is an important Python tool for doing numerical calculations. It has many mathematical methods that can work with big arrays and matrices with many dimensions. It was used for handling data structures and doing math tasks in this project.
4. **Scikit-learn:** It is a powerful tool for running machine learning tasks. It includes resources for choosing models, training them, and assessing their performance. Linear Regression, K-Nearest Neighbors, Decision Trees, and Random Forests are some of the models that are used in this project. Furthermore, it includes capabilities for hyperparameter configuration and cross-validation (GridSearchCV).
5. **Matplotlib and Seaborn:** Matplotlib gives you a strong base for making static, dynamic, and interactive plots, and Seaborn lets you make interesting and useful statistical graphics. Libraries for data display show association vectors, model success measures, and feature ranges.
6. **Jupyter Notebook:** Jupyter Notebook is a web-based platform for computing where users may build and share documents with narrative prose, visualizations, live code, and more. As part of this project, it is used for clearly describing the analytical method, writing code, and presenting the results.
7. **XGBoost:** One quick and adaptable variant of gradient boosting is XGBoost. It has made a name for itself thanks to its precision and effectiveness in machine learning tasks. The project's investigators tested its ability to process large data and provide predictions.

Methodology:

A. Data Collection:

Using pandas' read_csv method to load the train-data.csv CSV file was how the information was gathered. The first look at the dataset with ‘df.head()’ showed the structure of the data and the first few lines.

B. Data Preparation:

The dataset was pulled from a CSV file, and ‘df.info()’ was used to look at its general layout. The null values across each column were assessed using

'df.isnull().sum()'. This analysis helped find any missing data that had to be filled in before the next step could be taken. The distribution of the objective variable, "Price," was shown via a graph that used a KDE (Kernel Density Estimate). The graph showed where and how often car prices showed up in the data.

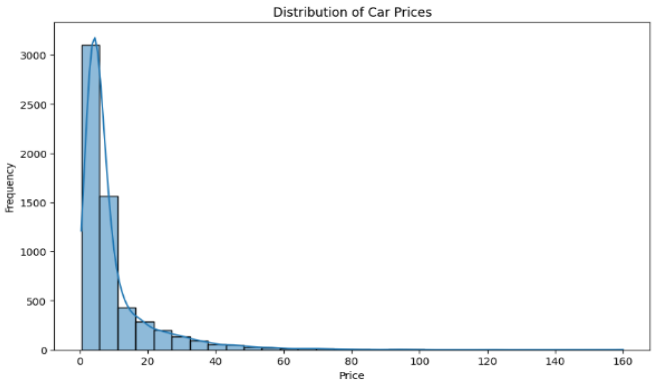

Figure 1: Figure shows where and how often car prices showed up.

I used 'df.describe()' to find descriptive statistics for numerical features. These statistics showed the center trends, spread, and shape of the dataset's distribution. I found the amount of lost data for each column and showed it. For the Price variable, a box plot was made to look for any possible exceptions. The following image helped find numbers that were too high or too low for no reason, which could affect how well the model worked.

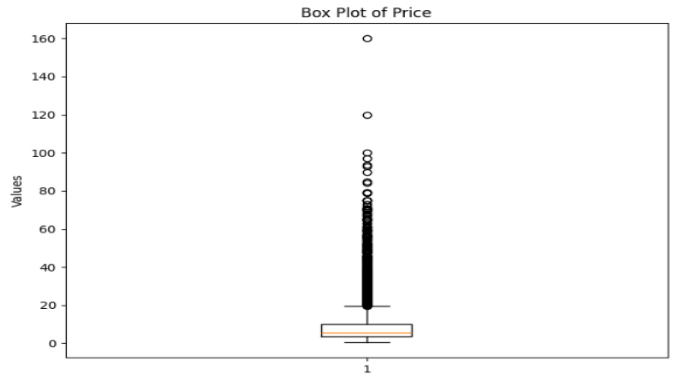

Figure 2: Figure helps find numbers that were too high or too low for no reason.

C. Exploratory Data Analysis:

To clean up the information and focus on the important traits, columns that weren't needed were taken out. A lot of sections that had "Unnamed" in their names were removed. To make sure the data was full, rows that were missing numbers in important fields like "Mileage," "Engine," "Power," and "Seats" were taken out. There were more than 85% blanks in the "New_Price" field, so it was also removed. The remaining dataset was updated, and the missing data percentage was calculated. We found and got rid of any outliers in the "Price" column.

A box plot was used to see and prove that the outliers had been removed, giving a more accurate picture of how the data was distributed.

A histogram was then used to look at the frequency distribution of "Price" again and see what happened to the data's distribution after the outliers were taken out.

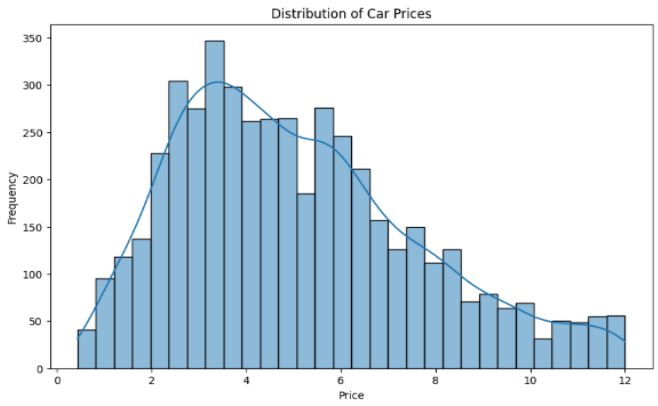

Figure 3: Frequency distribution of price.

To see how the unique numbers were spread out, I used count plots to keep track of them. This helped me figure out the dataset's group traits. This helped us figure out how common and different the groups were in the information. As an example, the

following graph shows how the car listings are distributed by "Fuel_Type."

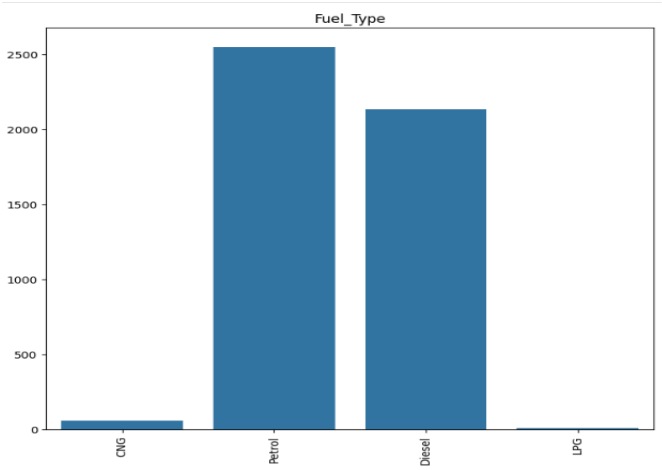

Figure 4: Car listings distributed by 'Fuel_Type'.

Boxplots were used to show how the goal variable "Price" was spread out across the different category features, but not across the "Name" column. Each category trait's relationship to "Price" was examined. In this method, the costs of various categories, such as region and fuel type, may be shown. Shown in the image below is the distribution of pricing by 'Location' as an example.

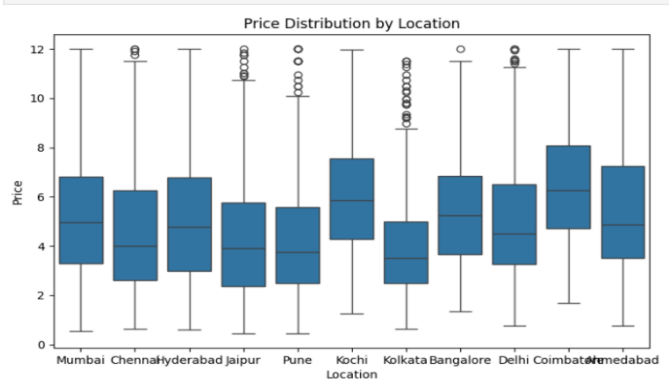

Figure 5: Distribution of pricing by 'Location'.

Scatter plots were used to determine the relationship between "Price" and numerical attributes. This had elements such as "Year," "Kilometers_Driven," "Mileage," "Engine," "Power," and "Seats." The price is affected by various elements, and the graphs assist to identify patterns and correlations. To facilitate picture comparison, a grid was used. The connection between "Price" and other number factors, like "Year," can be seen in the below graph:

Figure 6: Connection between price factor and year factor.

An association image was built to reveal the interconnections between numerical characteristics. I was able to find relevant characteristics that may affect the model's performance by doing this

D. Encoding:

Getting category data ready for encoding means changing it into forms that machine learning systems can use. As part of this project, I use numbers to show category traits like "Name," "Location," "Fuel_Type," "Transmission," and "Owner_Type." The LabelEncoder is used to give each group in these features its unique number value. The LabelEncoder turns category columns into number data so that machine learning algorithms can understand and rate these characteristics. To provide correct analysis and predictions, it is important to encode category factors so that they can be easily added to the model training process.

E. Model Implementation and Results:

I changed the "fit_intercept" hyperparameter from True to False for the linear regression test. This simple linear model has an R2 value of 0.6969, an RMSE of 1.4153, and an MAE of 1.0672 for the training data. A little drop in R2 to 0.6934, 1.4215 RMSE, and 1.0710 MAE was observed on the test set. These results show that Linear Regression fits the data pretty well, but not as well as models with more complex features.

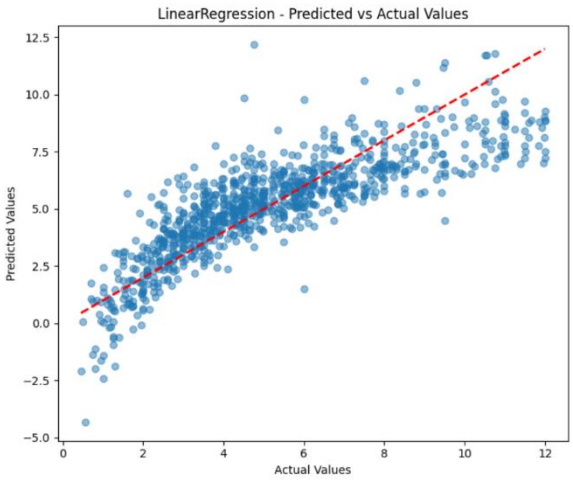

Figure 7: Predicted vs actual values for linear regression, illustrating the model's fit and the observed deviation.

Ridge Regression was tried with different solutions and different values for the regularization parameter alpha. With a training R^2 of 0.6968 and an RMSE of 1.4153, the model did about the same as Linear Regression. In the test set, the test R^2 was 0.6934, the RMSE was 1.4214, and the MAE was 1.0711. Ridge Regression was slightly better at dealing with overfitting, but compared to the easier Linear Regression, it wasn't really any better.

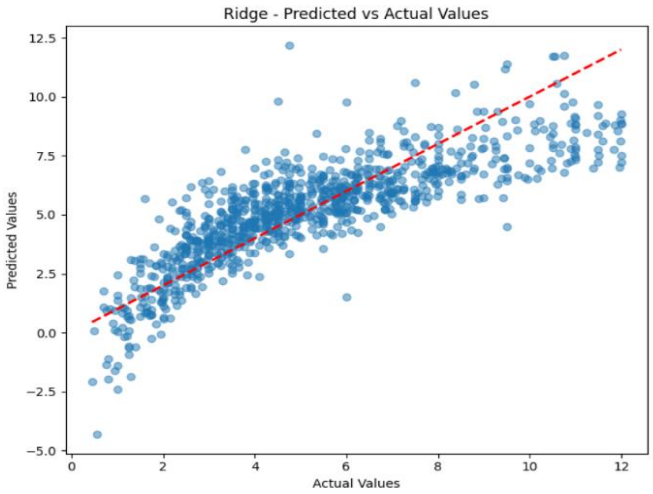

Figure 8: Predicted vs actual values for Ridge regression, showing a comparable fit to the simple linear regression model.

The K-Nearest Neighbors (KNN) Regressor was set up with various distance measures, distance weights, and the number of neighbors. A perfect training R^2 of 1000000, a very low RMSE of 0.0157, and a very low MAE of 0.0007 were all achieved by this model. But this perfect success in training points to overfitting. It had a test R^2 of 0.7022, an RMSE of 1.4013, and an MAE of 0.9762 on the test data. Even though there was a worry about overfitting, KNN did well and was competitive with the other models that were tried.

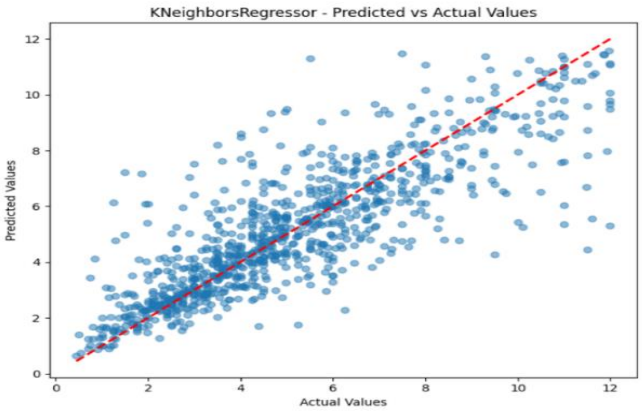

Figure 9: Predicted vs actual values for K-Nearest Neighbors Regressor, highlighting strong performance despite concerns of overfitting.

Different values for tree depth, minimum samples for splits, and maximum features were used to fine-tune the Decision Tree Regressor. Based on the training data, the model did very well, with a R^2 of 0.9006, an RMSE of 0.8104, and an MAE of 0.5790. It did a little worse on the test data, though, with a test R^2 of 0.7798, RMSE of 1.2049, and MAE of 0.8464. The Decision Tree Regressor did a great job of fitting the training data, but its test results show that it may overfit sometimes. It is still better than simpler models, though.

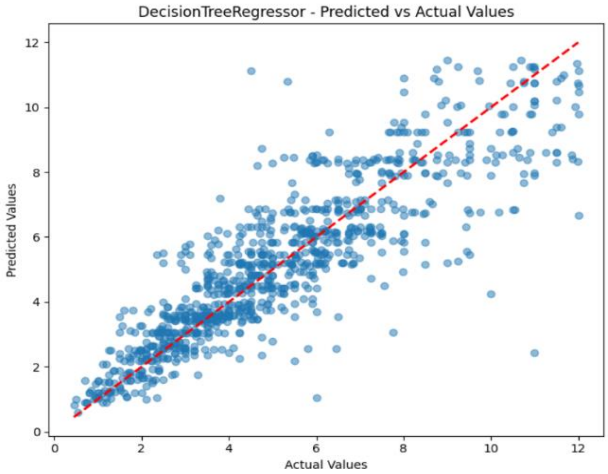

Figure 10: Predicted vs actual values for Decision Tree Regressor, highlighting strong performance on training data but potential overfitting on test data.

Random Forest Regressor was tested with different tree counts, depths, and feature selections. This model fared best with a training R^2 of 0.9818, RMSE of 0.3469, and MAE of 0.2441. It scored the greatest R^2 of 0.8747, lowest RMSE of 0.9092, and lowest MAE of 0.6456 on the test set. The Random Forest Regressor performed best across all criteria, demonstrating it catches data patterns and generalizes well to new data.

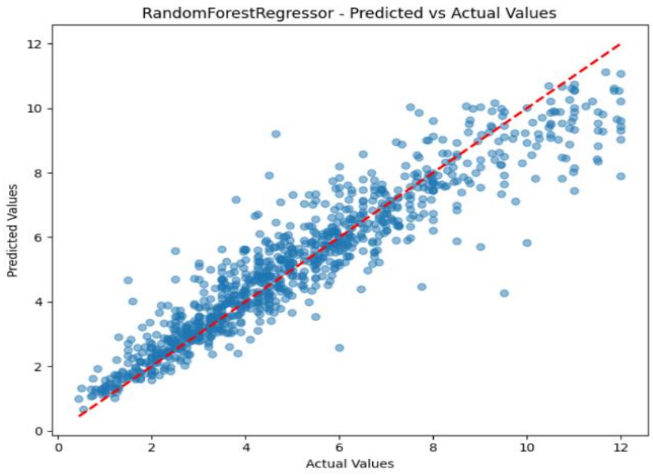

Figure 11: Predicted vs actual values for Random Forest Regressor, demonstrating excellent performance with strong pattern recognition and generalization to new data.

F. Combined Results:

The code checks and describes how well different machine learning models work across a number of cross-validation folds. For each model over all folds, metrics like R2, RMSE, and MAE are produced by running 'df_combined_results,' which gives a complete view of performance. By calculating the mean and standard deviation over all folds, the second action, 'df_results,' allows for a comparison of the efficacy and consistency of each model. "df_combined_results" shows the results for each fold to check how stable the model is, while "df_results" combines these results to show which model did exceptionally well based on all the measures.

	Model	Mean Train R^2	Std Train R^2	Mean Test R^2	Std Test R^2
0	LinearRegression	0.696853	0.006087	0.693405	0.023488
1	Ridge	0.696850	0.006087	0.693434	0.023393
2	KNeighborsRegressor	0.999961	0.000017	0.702158	0.015336
3	DecisionTreeRegressor	0.900578	0.003312	0.779807	0.015900
4	RandomForestRegressor	0.981788	0.000413	0.874680	0.006638

Figure 11: This table shows how the mean R^2 values of all the models for both training and test sets.

G. Conclusion

Based on test R^2, RMSE, and MAE, the RandomForestRegressor was determined to be the first-rate model. Because it can manage huge datasets and pick out complicated trends, it can be used to make very accurate predictions about car prices. Even though the alternative models had been useful, they were not as good as the RandomForestRegressor at making predictions and decreasing errors. The DecisionTreeRegressor also had precise effects, but it was more likely to fit too well than the RandomForestRegressor. Even though the KNeighborsRegressor did good in training, it no longer did plenty better on check data than the RandomForestRegressor. The LinearRegression and Ridge models gave a good starting point but weren't as good at describing the dataset's complexity.

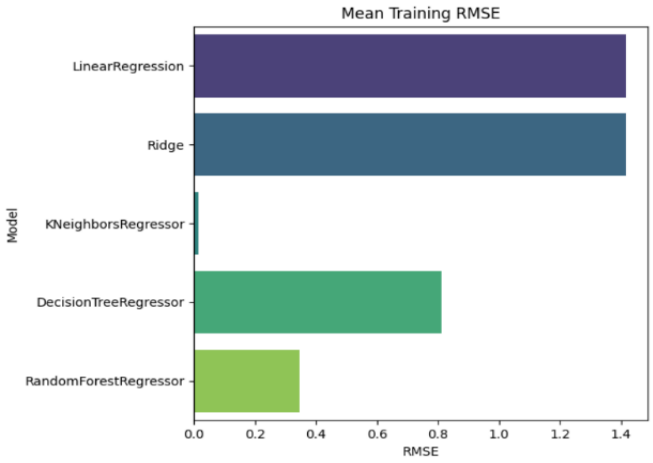

Figure 12: Mean Training RMSE comparison across models, with Random Forest Regressor achieving the lowest RMSE, indicating superior performance during training.

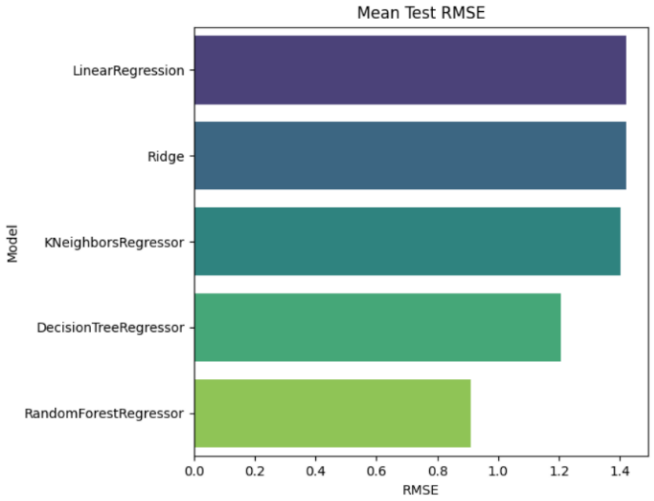

Figure 13: Mean Test RMSE comparison across models, with Random Forest Regressor achieving the lowest RMSE, indicating superior performance during test.

H. Future Work

In the future, researchers will try to improve the model by adding more complex methods, like Neural Networks, and things like brand image and market trends. To make things even more accurate and useful, hyperparameters will be find-tuned and a real-time release process will be put in place. Upcoming changes will also include information on hybrid, electric, and self-driving cars to make sure the system stays current with how quickly automobile technology is changing.

References:

- Samruddhi, K. and Kumar, R.A., 2020. Used car price prediction using K-nearest neighbor-based model. *Int. J. Innov. Res. Appl. Sci. Eng. (IJIRASE)*, *4*(3), pp.2020-686.
- Gegic, E., Isakovic, B., Keco, D., Masetic, Z. and Kevric, J., 2019. Car price prediction using machine learning techniques. *TEM Journal*, *8*(1), p.113.
- Noor, K. and Jan, S., 2017. Vehicle price prediction system using machine learning techniques. *International Journal of Computer Applications*, *167*(9), pp.27-31.
- Kuiper, S., 2008. Introduction to multiple regression: How much is your car worth?. *Journal of Statistics Education*, *16*(3).
- Fathalla, A., Salah, A., Li, K., Li, K. and Francesco, P., 2020. Deep end-to-end learning for price prediction of second-hand items. *Knowledge and Information Systems*, *62*(12), pp.4541-4568.
- Zhang, Y.S., 2018. A Used Cars' Price Forecasting Model Based on Artificial Neural Network. *Tianjin University: Tianjin, China*.
- Chen, S. and Liu, Z., 2022, January. Application of data mining technology in second-hand car price forecasting. In *2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI)* (pp. 260-273). IEEE.
- Jin, C., 2021, November. Price prediction of used cars using machine learning. In *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)* (pp. 223-230). IEEE.
- Dietterich, T., 1995. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, *27*(3), pp.326-327.