Srija Kota and Pavani Ayanambakam – CSE 587

Project Phase III Report

Abstract:

The primary objective of our project is to provide accurate price predictions. These predictions will assist sellers and buyers in determining the market value of a used vehicle, enabling them to make informed decisions about potential profits or losses. Our project aims to facilitate vehicle exchange and sales without unexpected losses. The dataset is chosen, and Exploratory Data Analysis is performed using statistical and graphical methods as this is the crucial step for feature selection, modeling, and interpretation of results. Out of 6 machine learning models we built, the K Nearest Neighbors (KNN) model demonstrated superior performance and was selected for predicting used vehicle prices. The performance of the models was evaluated using metrics like Mean Squared Error, accuracy, R2 score and precision. The successful implementation of the KNN model and the GUI interface provides a practical solution for users seeking accurate and data-driven predictions for selling their used vehicles. A user interface for price prediction is built that allows users to input relevant information about a used vehicle, and the application predicts the price based on the provided details. HTML, ngrok and Flask are used to build and deploy our user interface and the model. Our product provides seamless and enjoyable experience for users while ensuring the accuracy and transparency of the price prediction process.

Problem Statement:

Our project aims to find a suitable price for selling used automobiles/vehicles by analyzing and considering all their features, such as year of registration, miles driven, model, gear box type, and more. We seek to identify the factors or variables that significantly impact the process of predicting the price.

Data Sources

The primary dataset used for this analysis was from Kaggle. It contains comprehensive information about used automobile listings, including details such as vehicle specifications, seller information, and more. The specific dataset used for this project can be accessed via the following link:

https://www.kaggle.com/datasets/thedevastator/uncovering-factors-that-affect-used-car-prices/data

Usage of Models

For the Used Automobiles Price prediction project we built 6 Machine Learning models. They are:

- Linear Regression
- Polynomial Regression
- KNN model
- Random Forest
- Gradient Boosting
- Bayesian Ridge regression

The performance of the above models is evaluated using MSE (Mean Squared Error), R2-score.

We ended up using KNN (K Nearest neighbors) model among all for predicting the price of used cars. We picked the features (independent variables) from the dataset to predict the price. We chose the features that have a significant impact on determining the price of used vehicles like powerPS, kilometer, vehicleType, model, gearbox type, abtest type and if there is any damage. Later we decided on the number of neighbors (K) to consider when making predictions. This was determined through hyperparameter tuning and cross-validation. The distance metric (Euclidean distance) is used to measure the similarity between datapoints. With the selected features, K value, and distance metric we trained the KNN regression model on the training dataset.

Hyper Parameter Tuning

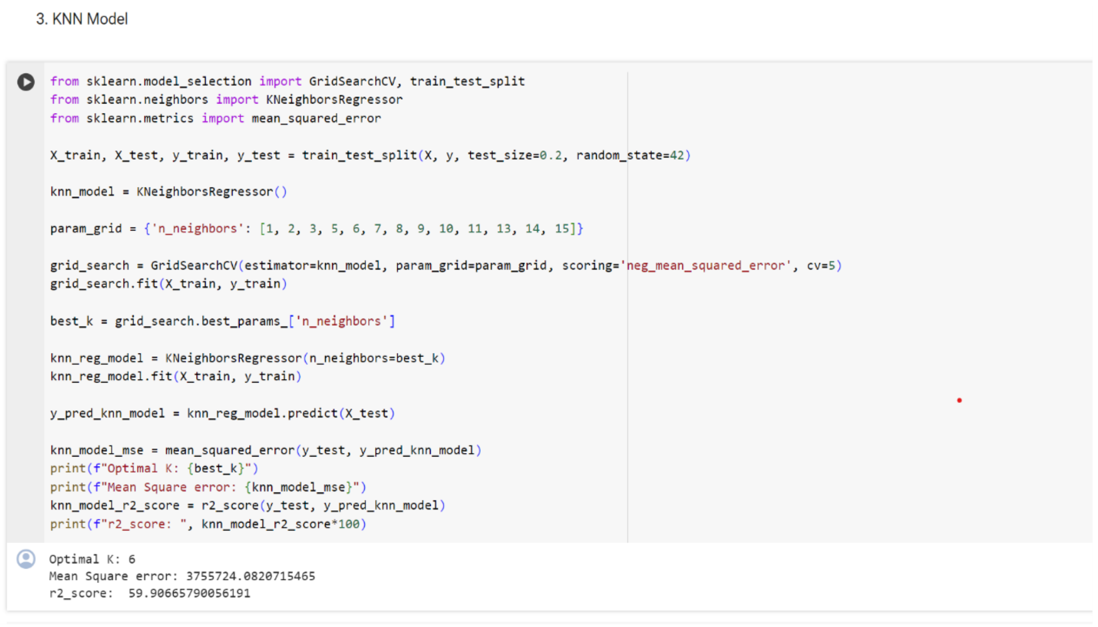
For each instance in the test set, we used the trained KNN model to predict the price based on the K nearest neighbors in the training set. For KNN, the primary hyperparameter is the number of neighbors (K). So, we conducted hyperparameter tuning using techniques like grid search to find the optimal value for K. Firstly, we specified a range of values for the hyperparameters we wanted to tune. Using the GridSearchCV class from scikit-learn, we set up a grid search. The key parameters are as follows:

estimator: The base model, which is the KNeighborsRegressor in our case.
param_grid: The grid of hyperparameter values to search through.
cv: The number of cross-validation folds. Here, cv=5 indicates 5-fold cross-validation.
scoring: The evaluation metric used to determine the best hyperparameters. We used 'neg_mean_squared_error' and the negative sign indicates that the scoring function returns values to be maximized.
After the grid search was completed, we retrieved the best hyperparameter values. In this case, we extracted the optimal number of neighbors (best_k) from the best_params attribute of the GridSearchCV object. Finally, we created the KNearestNeighbors model using the optimal number of neighbors obtained from the grid search. This model knn_model is considered the final model with the best hyperparameter configuration based on the specified grid.

We evaluated the performance of the model using regression metrics such as Mean Squared Error (MSE), R-squared. These metrics quantify how well the predicted prices align with the actual prices. We achieved an r2_score of 59.9.

Below is the screenshot of the code block and output of the KNN Model:

Srija Kota and Pavani Ayanambakam – CSE 587


3. KNN Model

```
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn_model = KNeighborsRegressor()

param_grid = {'n_neighbors': [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]}

grid_search = GridSearchCV(estimator=knn_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train, y_train)

best_k = grid_search.best_params_['n_neighbors']

knn_reg_model = KNeighborsRegressor(n_neighbors=best_k)
knn_reg_model.fit(X_train, y_train)

y_pred_knn_model = knn_reg_model.predict(X_test)

knn_model_mse = mean_squared_error(y_test, y_pred_knn_model)
print(f"Optimal K: {best_k}")
print(f"Mean Square error: {knn_model_mse}")
knn_model_r2_score = r2_score(y_test, y_pred_knn_model)
print(f"r2_score: ", knn_model_r2_score*100)
```

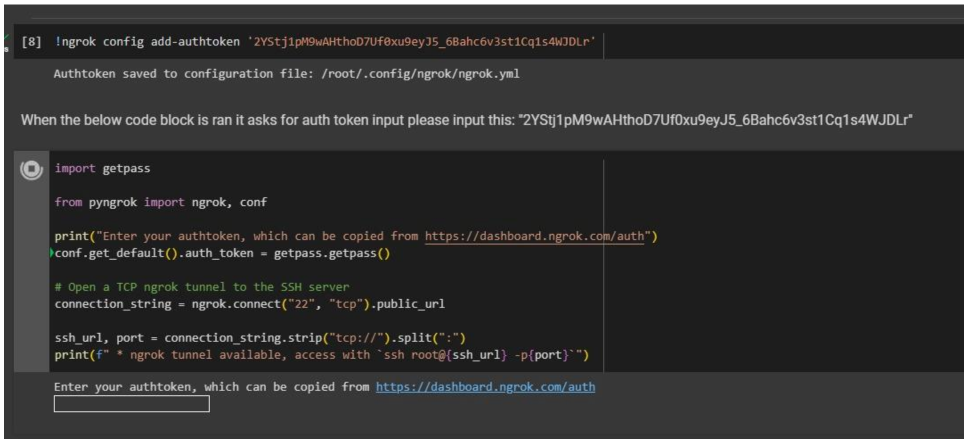
```
Optimal K: 6
Mean Square error: 3755724.0820715465
r2_score:  59.90665790056191
```

Deployment of our model and running of the application:

A web application is developed for taking the input details of from the user like model, brand, mileage, number of kilometers travelled, and so on. Two HTML pages were created for structuring the content on the web. One page for taking input from the user and one for displaying the predicted price and useful insights to the user.

Before running the phase 3 code create a folder called "static" and upload the images "prediction_car.jpeg","car_home_page.jpeg","graph-1.jpeg","graph-2.jpeg","graph-3.jpeg","graph-4.jpeg" into it. Create another folder called "templates" and upload the "index.html" and "predicted.html" files into it. Also add the pickle file "knn_pickle_file" to load the model.

For running the Flask application, we need an AUTH token. So, we generated it on a ngrok website. While running the code, it asks for the AUTH token to be given as input as shown below. Enter it in the field and run the next block.



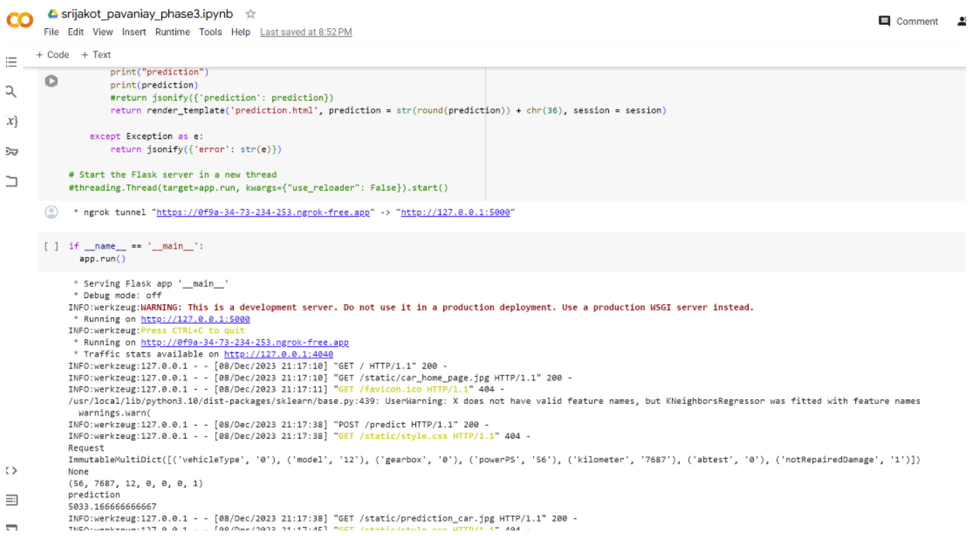
```
[8] !ngrok config add-authtoken '2YStj1pM9wAHthoD7Uf0xu9eyJ5_6Bahc6v3st1Cq1s4WJDLr'

    Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

When the below code block is ran it asks for auth token input please input this: "2YStj1pM9wAHthoD7Uf0xu9eyJ5_6Bahc6v3st1Cq1s4WJDLr"

```
import getpass

from pyngrok import ngrok, conf

print("Enter your authtoken, which can be copied from https://dashboard.ngrok.com/auth")
conf.get_default().auth_token = getpass.getpass()

# Open a TCP ngrok tunnel to the SSH server
connection_string = ngrok.connect("22", "tcp").public_url

ssh_url, port = connection_string.strip("tcp://").split(":")
print(f" * ngrok tunnel available, access with `ssh root@{ssh_url} -p{port}`")
```

```
Enter your authtoken, which can be copied from https://dashboard.ngrok.com/auth
```

The Flask development server will start and we can see output indicating that the server is running. Below is the screenshot which shows running of FLASK application successfully:


srijakot_pavaniay_phase3.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:52 PM
Comment
+ Code + Text

```
        print("prediction")
        print(prediction)
        #return jsonify({'prediction': prediction})
        return render_template('prediction.html', prediction = str(round(prediction)) + chr(36), session = session)

    except Exception as e:
        return jsonify({'error': str(e)})

# Start the Flask server in a new thread
#threading.Thread(target=app.run, kwargs={"use_reloader": False}).start()
```

```
 * ngrok tunnel "https://0f9a-34-73-234-253.ngrok-free.app" -> "http://127.0.0.1:5000"
```

```
if __name__ == '__main__':
  app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://0f9a-34-73-234-253.ngrok-free.app
 * Traffic stats available on http://127.0.0.1:4040
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:10] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:10] "GET /static/car_home_page.jpg HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:11] "GET /favicon.ico HTTP/1.1" 404 -
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsRegressor was fitted with feature names
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:38] "POST /predict HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:38] "GET /static/style.css HTTP/1.1" 404 -
Request
ImmutableMultiDict([('vehicleType', '0'), ('model', '12'), ('gearbox', '0'), ('powerPS', '56'), ('kilometer', '7687'), ('abtest', '0'), ('notRepairedDamage', '1')])
None
(56, 7687, 12, 0, 0, 0, 1)
prediction
5033.166666666667
INFO:werkzeug:127.0.0.1 - - [08/Dec/2023 21:17:38] "GET /static/prediction_car.jpg HTTP/1.1" 200 -
```

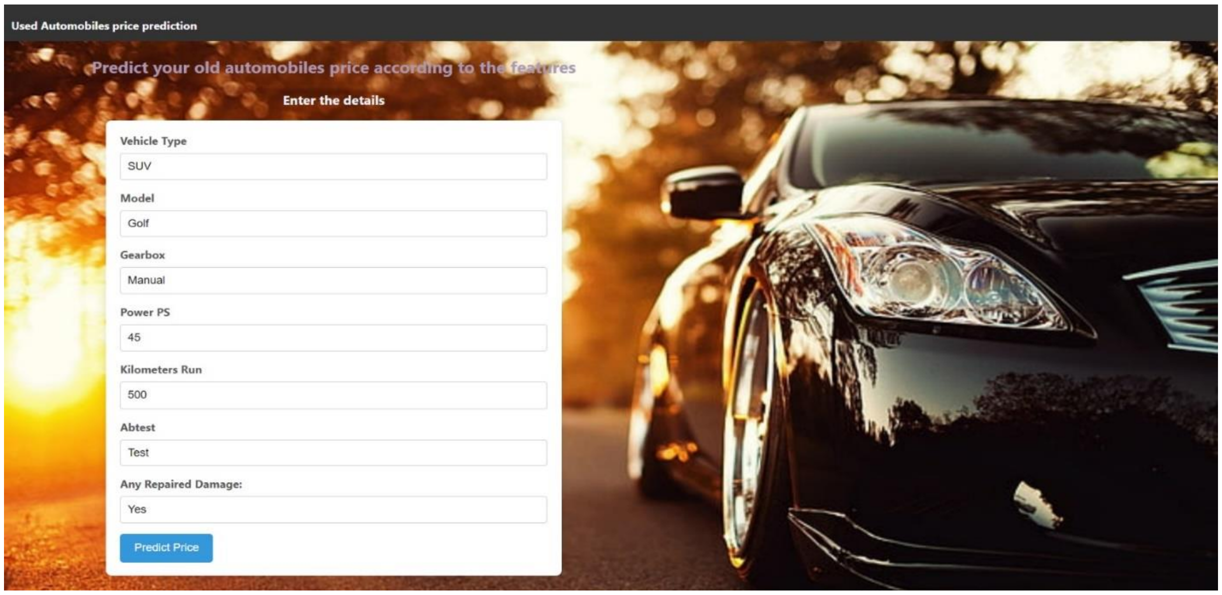
This application has routes for rendering HTML templates and handling a prediction endpoint. The starting page involves extracting input data from a form and passing it to a model for prediction. The result is then rendered on a separate prediction template. We created a new code block and imported the Flask module. Then we created an instance of the Flask class. We defined routes to specify how our application responds to different URLs. We passed data from the routes to the HTML templates and finally the predicted price and the current dynamics are displayed.

Working Instructions to use our product

An input form (Used Automobiles price prediction) is provided to the user asking for the details of the automobile. The input fields are vehicle type, it's model, type of gearbox, power PS, number of kilometers travelled, Ab test, any repaired damages.

Below is the screenshot of the UI of the landing page with the input fields to be filled:

Srija Kota and Pavani Ayanambakam – CSE 587


Used Automobiles price prediction

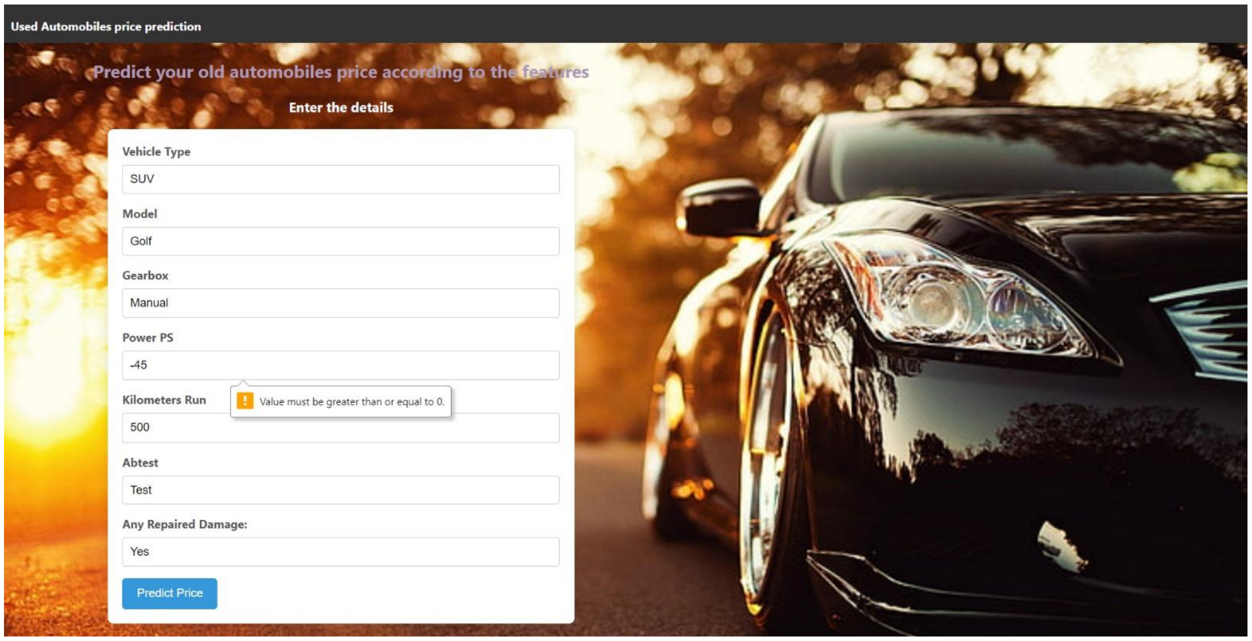
Predict your old automobiles price according to the features

Enter the details

Vehicle Type	SUV
Model	Golf
Gearbox	Manual
Power PS	45
Kilometers Run	500
Abtest	Test
Any Repaired Damage:	Yes

We have added input validations to the above form as they help users understand what is expected in each form field. They serve as guidance, indicating the format, type, or range of acceptable input. Clear and descriptive messages help users to correct errors and complete the form accurately. Validation messages in forms are crucial for providing feedback to users about the correctness of the data they've entered. For example, when an unacceptable value like -45 is given which is negative to Power PS field it throws an error.

Below is the screenshot of the input fields with error validation:


Used Automobiles price prediction

Predict your old automobiles price according to the features

Enter the details

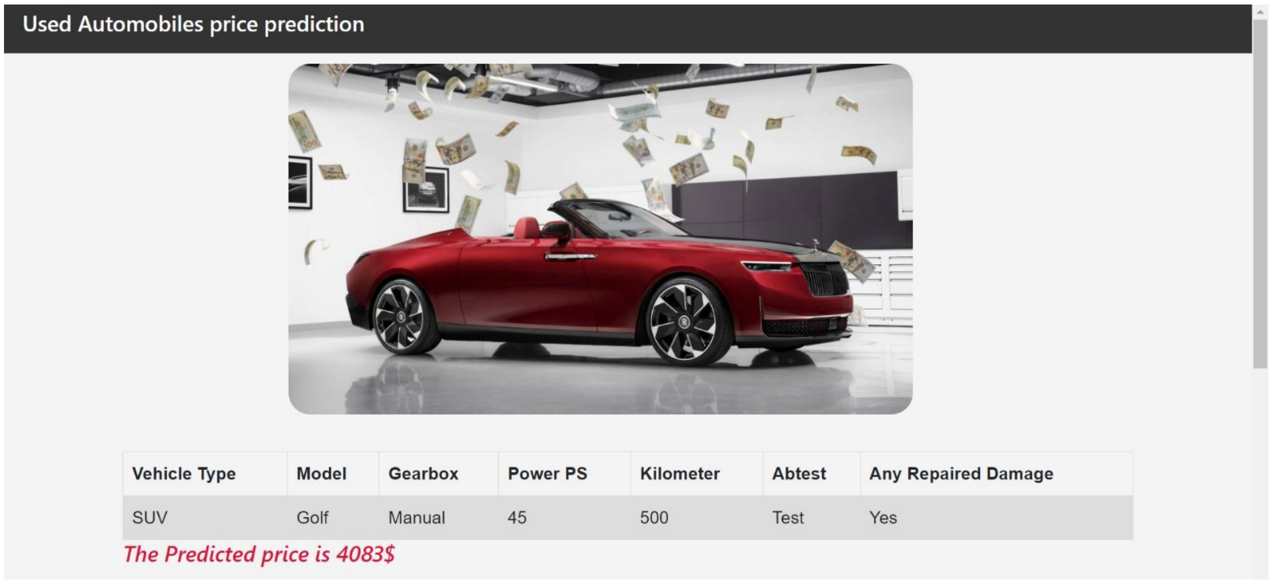
Vehicle Type	SUV
Model	Golf
Gearbox	Manual
Power PS	-45
Kilometers Run	500
Abtest	Test
Any Repaired Damage:	Yes

Value must be greater than or equal to 0.

Once the user enters the valid details and clicks the button "Predict Price", a page will be redirected which displays the used automobile's predicted price in the current market.

Srija Kota and Pavani Ayanambakam – CSE 587

Below is the screenshot of the UI of the predicted screen which displays the predicted price in highlighted color and also the details entered by the user in the input form.



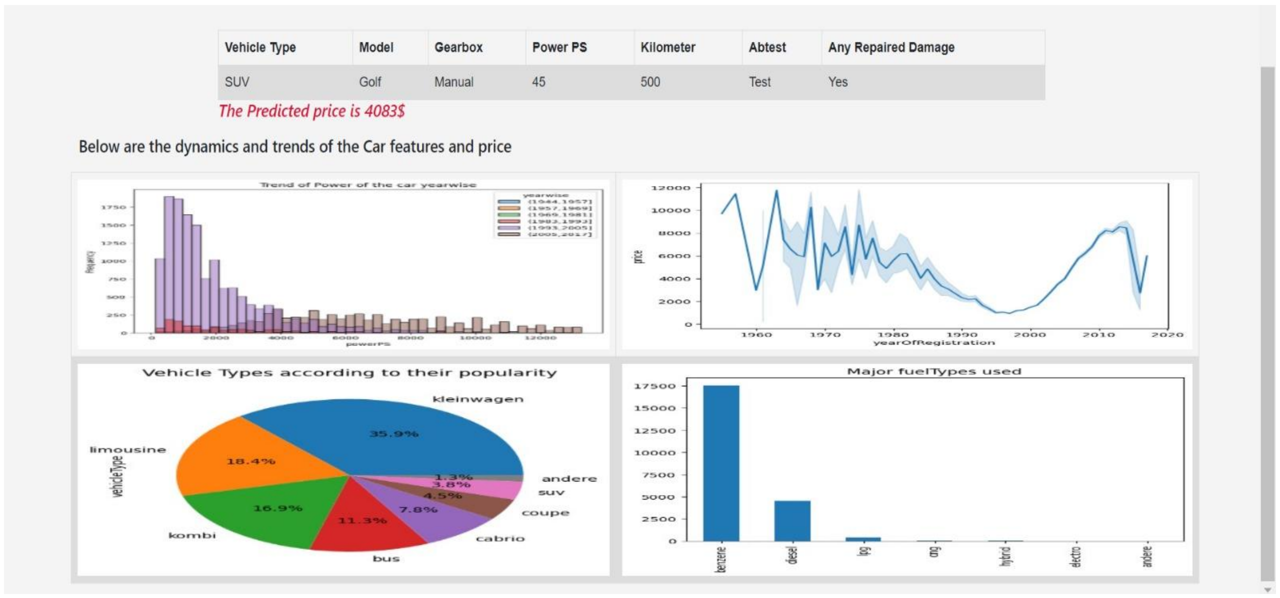
Used Automobiles price prediction

Vehicle Type	Model	Gearbox	Power PS	Kilometer	Abtest	Any Repaired Damage
SUV	Golf	Manual	45	500	Test	Yes

The Predicted price is 4083$

This page in addition also displays relevant visualization graphs (dynamics of the varying price) that will be helpful for user in getting the idea of used automobile's price based on the different features like kilometers it travelled, brand, model and so on.

Below is the screenshot of the page displaying the visualization graphs:



Vehicle Type	Model	Gearbox	Power PS	Kilometer	Abtest	Any Repaired Damage
SUV	Golf	Manual	45	500	Test	Yes

The Predicted price is 4083$

Below are the dynamics and trends of the Car features and price

Srija Kota and Pavani Ayanambakam – CSE 587

Recommendations based on our analysis

Our analysis from the price prediction problem provides users with insights into the used car market. This could include market trends, popular car models, or geographical variations in pricing. Such insights can be valuable for both buyers and sellers.

In our project we implemented a dynamic pricing strategy that considers real-time market conditions, demand-supply dynamics, and other external factors to provide more accurate and dynamic price predictions. We develop a user-friendly interface or application that allows users to input details about a used car like kilometers it travelled, geartype, it's model, power PS, abtest and other details. Then our model provides a predicted price. This could be beneficial for individuals who are looking to buy or sell used cars. We have also implemented interactive visualization to help users explore trends and patterns in the dataset. Visualizations could include price distributions, brand comparisons, or geographical insights.

Our project can be extended by integrating it with online platforms. We can integrate our model with online platforms like KelleyBlueBook, eBay, Edmunds etc., or marketplaces to offer users a seamless experience in obtaining price predictions directly when listing or searching for used cars. We can also explore the integration of additional relevant data sources, such as economic indicators, consumer sentiment, or specific events affecting the used car market. Other extension to our project can be collecting user feedback and iterating over the model and features based on user suggestions and needs. Continuous improvement is essential for creating a valuable and user-centric product.
Our project can offer valuable insights into the factors that influence vehicle rescale values. This information will help market participants make more informed decisions and adapt to changing market dynamics. In the e-commerce platform, many marketing and sales applications are offered. Buying and selling used vehicles for profit is one of the popular businesses on this platform. Predicting these prices is a crucial aspect of our project, as it empowers users to set competitive prices for their vehicles in the market.

References:

1. https://flask.palletsprojects.com/en/3.0.x/
2. https://www.w3schools.com/
3. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
4. https://pythonbasics.org/deploy-flask-app/