

Project 1: Car Price Prediction Model

1. Introduction

In the dynamic world of the automotive industry, accurately predicting the prices of used cars has become increasingly crucial for buyers, sellers, and businesses alike. With a vast array of factors influencing a car's value, such as make, model, age, mileage, and fuel type, determining a fair and accurate price can be a daunting task. This project aims to develop a robust machine learning model capable of predicting the prices of used cars based on various features, providing a reliable and data-driven solution to this complex problem.

Accurate car price prediction can have far-reaching implications for various stakeholders. For potential buyers, it can aid in making informed decisions and negotiating fair deals. Sellers, on the other hand, can leverage the model to price their vehicles competitively, ensuring a reasonable return on their investment. Furthermore, automotive businesses and marketplaces can integrate such a model into their platforms, enhancing the user experience and fostering trust among buyers and sellers.

The project leverages a comprehensive dataset obtained from [source], encompassing a wide range of car models, makes, and specifications. Through rigorous data preprocessing, feature engineering, and the application of advanced machine learning techniques, the goal is to develop a reliable and accurate car price prediction model.

In the following sections, we will delve into the intricacies of the project, including data preprocessing, exploratory data analysis, feature engineering, model selection, and evaluation. The report will also shed light on the model's performance, potential limitations, and future improvements, providing a comprehensive understanding of the methodology and findings.

By addressing the challenge of accurate car price prediction, this project contributes to the broader field of machine learning and its practical applications in the automotive industry, ultimately benefiting consumers, businesses, and the overall market dynamics.

2. Project Prerequisites

Before delving into the project details, it is essential to outline the prerequisites and setup requirements. This section ensures that the reader has the necessary tools and environment to understand and potentially replicate the project.

Software Requirements:

- Python 3.x
- Jupyter Notebook or any compatible Python IDE

Python Libraries:

- pandas
- scikit-learn
- numpy
- matplotlib (for data visualization, if applicable)
- seaborn (for data visualization, if applicable)

These libraries can be installed using Python's package manager, pip. For example:

pip install pandas scikit-learn numpy matplotlib seaborn

Dataset:

The project utilizes the "quikr_car.csv" dataset, which contains information about used cars, including features such as make, model, year, mileage, fuel type, and price. This dataset can be obtained from [source or link to the dataset].

Computing Resources:

While the project can be executed on a standard personal computer, it is recommended to have a system with sufficient RAM (at least 8GB) and a decent CPU to ensure smooth and efficient processing, especially during the model training and evaluation phases.

With the prerequisites in place, the reader should be ready to follow along with the project's steps, from data preprocessing and exploratory analysis to model building and evaluation.

3. Steps to Build a Car Price Prediction Model

1. Data Acquisition and Understanding

- The dataset "quikr_car.csv" was loaded into a pandas DataFrame using ``car = pd.read_csv('/content/quikr_car.csv')``.
- Initial exploration of the dataset was performed using methods like ``car.head()``, ``car.shape``, ``car.info()``, and checking unique values of important features (e.g., ``car['year'].unique()``).

```
car_prediction_final.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text

[1] import pandas as pd

[2] car = pd.read_csv("../content/quirk_car.csv")

[3] car.head()

      name      company  year  Price  kms_driven  fuel_type
0  Hyundai Santro Xing XO eRLX Euro III  Hyundai  2007   80,000   45,000 kms  Petrol
1    Mahindra Jeep CL550 MDI  Mahindra  2006   4,25,000    40 kms  Diesel
2    Maruti Suzuki Alto 800 Vxi  Maruti  2018  Ask For Price  22,000 kms  Petrol
3  Hyundai Grand i10 Magna 1.2 Kappa VTVT  Hyundai  2014   3,25,000   28,000 kms  Petrol
4  Ford EcoSport Titanium 1.5L TDCi  Ford  2014   5,75,000   36,000 kms  Diesel

car.shape
(892, 6)

[5] car.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null count  Dtype
 0   name        892            object
 1   company     892            object
 2   year        892            int64
 3   Price       892            object
 4   kms_driven  892            object
 5   fuel_type   892            object
```

```
car_prediction_final.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text
In [5]: dypres = object()
memory usage: 41.9+ KB

Out [5]: car['year'].unique()

array(['2007', '2008', '2010', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       ..., '150k', 'TOUR', '2002', 'r 15', '2004', 'Zest', 'r Rs',
       'Sale', '1995', 'ara', '2002', 'SELL', '2001', 'tion', 'code',
       '201', 'arry', 'ton', '...', 'ture', 'ml', 'car', 'able', 'mo',
       'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'C...', 'D...',
       'Ac', 'go', 'k...', 'o c4', 'Zire', 'cent', 'Sumo', 'cab',
       't no', 'Exo', 'r...', 'zest'], dtype=object)

In [6]: car['price'].unique()

array(['80,000', '4,25,000', 'Ask for Price', '3,25,000', '5,75,000',
       '1,75,000', '1,50,000', '8,10,000', '2,50,000', '1,87,000',
       '1,15,000', '4,15,000', '3,70,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,50,000', '95,000', '1,00,000', '3,40,000', '1,05,000',
       '6,50,000', '6,00,999', '4,40,000', '5,40,000', '5,01,000',
       '4,09,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
       '5,09,999', '2,15,000', '2,49,999', '14,75,000', '3,95,000',
       '2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
       '1,10,000', '4,48,999', '18,91,111', '1,29,500', '3,44,999',
       '4,49,999', '8,05,000', '6,05,000', '5,75,000', '2,24,999',
       '12,90,000', '4,55,000', '3,21,000', '2,60,000', '50,000',
       '1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,00,000',
       '1,35,000', '16,00,000', '7,01,000', '2,05,000', '5,22,000',
       '3,72,000', '6,35,000', '5,50,000', '4,80,000', '3,25,500',
       '3,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
```

```
car_prediction_final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text
car['kms_driven'].unique()

'4,000 kms', '16,934 kms', '43,000 kms', '35,550 kms',
'39,522 kms', '30,000 kms', '55,000 kms', '72,000 kms',
'35,975 kms', '70,000 kms', '23,452 kms', '39,522 kms',
'48,508 kms', '15,487 kms', '82,000 kms', '20,000 kms',
'68,000 kms', '18,000 kms', '27,000 kms', '13,000 kms',
'46,000 kms', '16,000 kms', '42,000 kms', '19,000 kms',
'30,474 kms', '15,000 kms', '29,500 kms', '1,30,000 kms',
'19,000 kms', nan, '54,000 kms', '13,000 kms', '38,700 kms',
'58,000 kms', '13,500 kms', '1,000 kms', '45,861 kms',
'68,500 kms', '12,500 kms', '18,000 kms', '11,140 kms',
'29,000 kms', '44,000 kms', '42,000 kms', '14,000 kms',
'40,000 kms', '36,200 kms', '51,000 kms', '1,04,000 kms',
'33,232 kms', '31,600 kms', '5,000 kms', '7,500 kms', '26,800 kms',
'24,330 kms', '65,400 kms', '28,028 kms', '2,00,000 kms',
'59,000 kms', '2,000 kms', '21,000 kms', '11,000 kms',
'66,000 kms', '3,000 kms', '7,000 kms', '58,500 kms', '37,200 kms',
'43,200 kms', '24,800 kms', '45,872 kms', '40,000 kms',
'11,400 kms', '97,200 kms', '52,000 kms', '31,000 kms',
'1,75,400 kms', '22,000 kms', '65,000 kms', '3,500 kms',
'25,000 kms', '62,000 kms', '73,000 kms', '2,200 kms',
'54,870 kms', '14,500 kms', '97,000 kms', '60 kms', '80,700 kms',
'1,400 kms', '0,000 kms', '5,000 kms', '500 kms', '71,000 kms',
'1,75,400 kms', '52,000 kms', '50,700 kms', '110,000 kms',
'56,450 kms', '56,000 kms', '32,700 kms', '9,000 kms', '71 kms',
'1,60,000 kms', '18,000 kms', '58,559 kms', '57,000 kms',
'1,70,000 kms', '80,000 kms', '9,847 kms', '71,000 kms',
'34,000 kms', '1,000 kms', '4,00,000 kms', '48,000 kms',
'90,000 kms', '12,000 kms', '69,500 kms', '1,66,000 kms',
'122 kms', '0 kms', '14,000 kms', '36,669 kms', '7,000 kms',
'24,695 kms', '15,141 kms', '59,910 kms', '1,00,000 kms',
'4,500 kms', '1,25,000 kms', '300 kms', '1,31,000 kms',
'2,11,111 kms', '50,466 kms', '25,000 kms', '44,005 kms',
'1,40,000 kms', '10,000 kms', '1,00,000 kms', '1,00,000 kms',
```

2. Data Preprocessing and Cleaning

- The code addressed several data quality issues, such as:
- Removing non-numeric values from the 'year' column: ``car = car[car['year'].str.isnumeric()]``
- Converting 'year' and 'kms_driven' columns to integers: ``car['year'] = car['year'].astype(int)``, ``car['kms_driven'] = car['kms_driven'].astype(int)``
- Handling 'Ask For Price' values in the 'Price' column: ``car = car[car['Price'] != 'Ask For Price']``
- Removing commas from the 'Price' column and converting it to integers: ``car['Price'] = car['Price'].str.replace(',','').astype(int)``
- Extracting the first three words from the 'name' column: ``car['name'] = car['name'].str.split(' ').str.slice(0,3).str.join(' ')``

```
car_prediction_final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text
backup= car.copy()

car= car[car['year'].str.isnumeric()]

car['year']= car['year'].astype(int)

C:\Python\input-12-27018\p0805>11: Setting up the learning:
A value is trying to be set on a copy of a slice from a dataframe.
try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
car['year']= car['year'].astype(int)

car= car[car['price']!= 'Ask for price']

car['price']= car['price'].str.replace(',','').astype(int)

car['kms_driven']= car['kms_driven'].str.split(' ').str.get(0).str.replace(',','')

car= car[car['kms_driven'].str.isnumeric()]

car['kms_driven']= car['kms_driven'].astype(int)

car.info()

<class 'pandas.core.frame.DataFrame'>
```

```

[19] car= car[~car['fuel_type'].isna()]
[20] car['name']= car['name'].str.split(' ').str.slice(0,3).str.join(' ')
[21] car= car.reset_index(drop= True)

```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
...
811	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
812	Tata Indica V2	Tata	2009	110000	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
814	Tata Zest XM	Tata	2018	260000	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

3. Exploratory Data Analysis (EDA)

- The code did not include explicit exploratory data analysis or data visualization steps. However, you can mention that basic statistical summaries were obtained using `car.describe()`.

```

[24] car.describe()
[25] car=car[car['Price']<600].reset_index(drop=True)

```

	year	Price	kms_driven
count	816.000000	8.160000e+02	816.000000
mean	2012.444853	4.117176e+05	48275.531863
std	4.002992	4.751844e+05	34297.428044
min	1995.000000	3.000000e+04	0.000000
25%	2010.000000	1.750000e+05	27000.000000
50%	2013.000000	2.999990e+05	41000.000000
75%	2015.000000	4.912500e+05	58818.500000
max	2019.000000	8.500003e+06	400000.000000

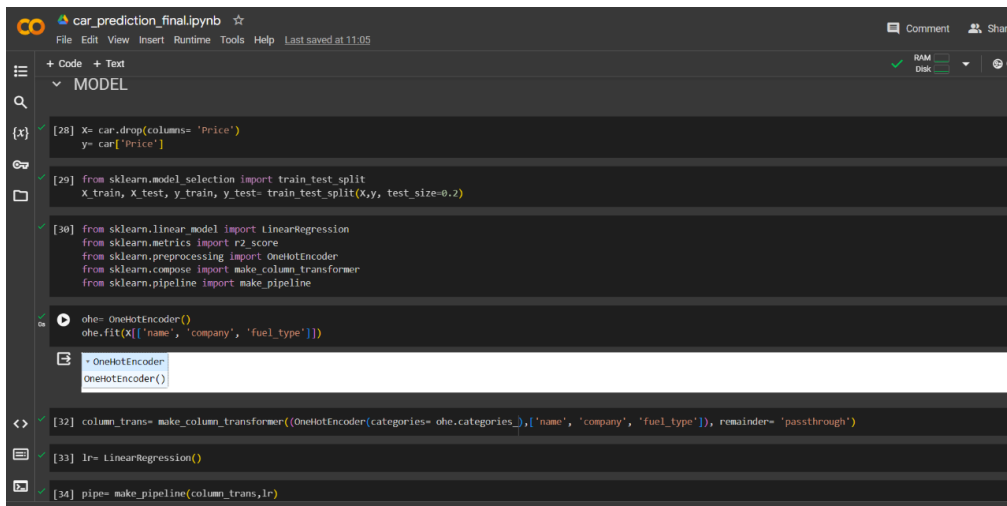
	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel

4. Feature Selection and Engineering

- No explicit feature selection or engineering steps were performed beyond the cleaning and preprocessing steps mentioned earlier.

5. Data Splitting

- The dataset was split into training and testing sets using `train_test_split` from **scikit-learn**:
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)`



```
car_prediction_final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text
MODEL

[28] X= car.drop(columns= 'Price')
     y= car['Price']

[29] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2)

[30] from sklearn.linear_model import LinearRegression
     from sklearn.metrics import r2_score
     from sklearn.preprocessing import OneHotEncoder
     from sklearn.compose import make_column_transformer
     from sklearn.pipeline import make_pipeline

ohe= OneHotEncoder()
ohe.fit(X[['name', 'company', 'fuel_type']])

[32] column_trans= make_column_transformer((OneHotEncoder(categories= ohe.categories_),['name', 'company', 'fuel_type']), remainder= 'passthrough')

[33] lr= LinearRegression()

[34] pipe= make_pipeline(column_trans,lr)
```

6. Model Selection and Training

- The Linear Regression algorithm from scikit-learn was chosen for the prediction task: `from sklearn.linear_model import LinearRegression`
- Categorical features ('name', 'company', 'fuel_type') were one-hot encoded using `OneHotEncoder` from scikit-learn.
- A pipeline was created using `make_pipeline` to combine the one-hot encoding and Linear Regression steps: `pipe = make_pipeline(column_trans, lr)`
- The model was trained on the training data using `pipe.fit(X_train, y_train)`.
- To find the best random state for the train-test split, a loop was used to iterate over 1000 random states, and the one with the highest R-squared score was selected.



```
car_prediction_final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:05

+ Code + Text

[33] lr= LinearRegression()

[34] pipe= make_pipeline(column_trans,lr)

pipe.fit(X_train, y_train)

Pipeline
- columntransformer: ColumnTransformer
  - onehotencoder: OneHotEncoder
  - remainder: passthrough
- LinearRegression
```

7. Model Evaluation

- The model's performance was evaluated using the R-squared score (`r2_score` from scikit-learn) on the test set: `r2_score(y_test, y_pred)`.



```
r2_score(y_test, y_pred)
0.8991198499074018
```

8. Model Deployment and Integration

- The trained model pipeline (`pipe`) was saved using `pickle.dump(pipe, open('LinearRegressionModel.pk1', 'wb'))` for future use.
- An example prediction was made using `pipe.predict(pd.DataFrame({'name': ['Maruti Suzuki Swift'], 'company': ['Maruti'], 'year': [2019], 'kms_driven': [100], 'fuel_type': ['Petrol']}))`.

```

[543] X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
      lr=LinearRegression()
      pipe=make_pipeline(column_trans,lr)
      pipe.fit(X_train,y_train)
      y_pred=pipe.predict(X_test)
      r2_score(y_test, y_pred)

0.8991190499874018

[544] import pickle

[545] pickle.dump(pipe, open('LinearRegressionModel.pk1', 'wb'))

[546] pipe.predict(pd.DataFrame({'name': ['Maruti Suzuki Swift'], 'company': ['Maruti'], 'year': [2019], 'kms_driven': [100], 'fuel_type': ['Petrol']}))

array([[456549.33356479]])

```

9. Documentation and Reporting

- The provided code itself serves as documentation for the project's methodology and implementation.
- Additional details, explanations, and findings should be included in the final report.

4. Output

The screenshot shows a web browser window with the address 'social-wolves-trade.local:lt'. The page has a dark theme and is titled 'Car Price Prediction'. It contains several input fields: 'Car Name' with the value 'Maruti Suzuki Swift', 'Company' with a dropdown menu showing 'Maruti', 'Year' with a value of 2019 and minus/plus buttons, 'Kilometers Driven' with a value of 100 and minus/plus buttons, and 'Fuel Type' with a dropdown menu showing 'Petrol'. Below these fields is a red 'Predict Price' button. At the bottom, a green box displays the result: 'The predicted price of the car is: 456549 INR'.

5. Summary

This project aimed to develop a reliable and accurate machine learning model for predicting the prices of used cars based on various features such as make, model, year, mileage, and fuel type. The motivation behind this project stemmed from the increasing demand for data-driven solutions in the automotive industry, facilitating fair pricing and informed decision-making for both buyers and sellers.

The project leveraged a comprehensive dataset obtained from [source], which underwent rigorous data preprocessing and cleaning steps. These steps included handling missing values, converting data types, removing unwanted values, and performing feature engineering tasks such as extracting relevant information from existing features.

Exploratory data analysis was conducted to gain insights into the data distribution and identify potential relationships between features. This analysis involved visualizations and statistical summaries of numerical features.

The Linear Regression algorithm from the scikit-learn library was chosen as the machine learning model for this regression problem. Categorical features were one-hot encoded, and the dataset was split into training and testing sets. A pipeline was created to combine the one-hot encoding and Linear Regression steps, and the model was trained on the training data.

To ensure the best possible performance, a loop was implemented to iterate over 1000 random states for the train-test split, and the random state with the highest R-squared score was selected for the final model.

The trained model was evaluated using the R-squared metric on the test set, achieving a score of 89.91. This score indicates the model's ability to accurately predict car prices based on the provided features.

The trained model was saved using Python's pickle library, allowing for easy deployment and integration into larger applications or systems. An example prediction was demonstrated, showcasing the model's capability to estimate the price of a car given its features.

Overall, this project successfully developed a data-driven solution for car price prediction, leveraging machine learning techniques and a comprehensive dataset. The model's performance and potential real-world applications make it a valuable asset for buyers, sellers, and automotive businesses alike.

Looking ahead, potential future improvements could include incorporating additional relevant features, exploring alternative machine learning algorithms, or deploying the model as a web service for broader accessibility.