USED CAR SELLING PRICE PREDICTION

Prepared By:

Aditya Muzumdar
Param Barge
Rupesh Bhalekar

Beautiful Soup

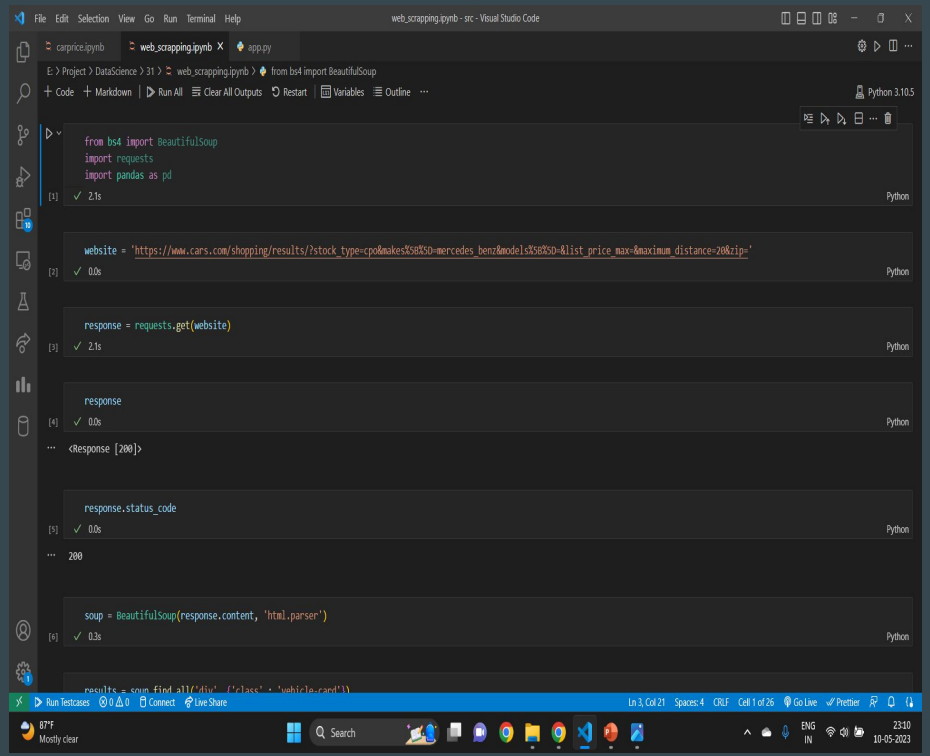
NumPy

scikit learn

Level 1 : Scraping Of Data

- The process of collecting and parsing raw data from the Web.
- Modules Use are : BeautifulSoup to Parse HTML and extract data from it.
- In this we have collected features like Car name,Year, selling price , present price,Kms_driven, fuel_type, transmission,Owner etc.
- Writing this raw extracted data into csv File.

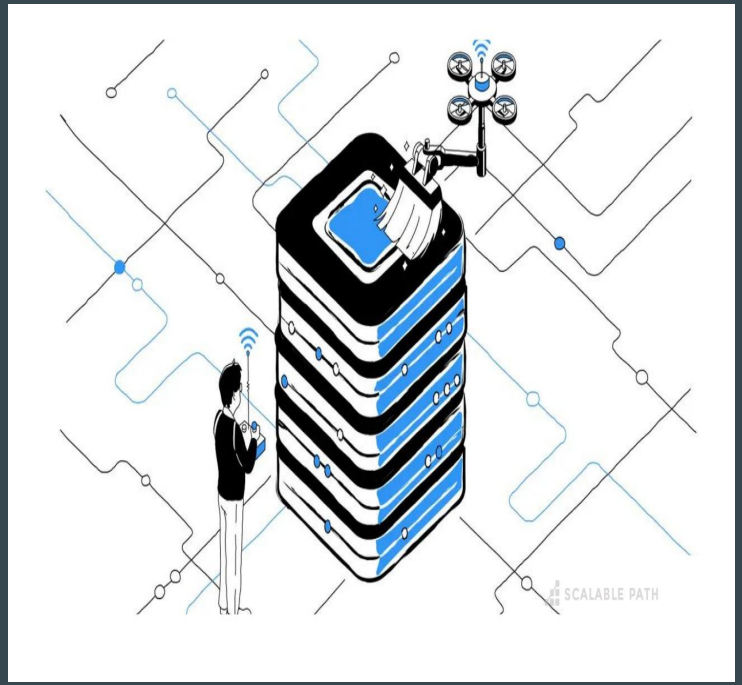
Level 1: Data Scraping



Car_Name	Year	Selling_Pri	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
Tata Nano Twist XT	2015	0.9	1.1	18,000	Petrol	Dealer	Manual	0
Hyundai Santro Xing GL	2010	1.22	1.42	46,733	Petrol	Dealer	Manual	0
Chevrolet Beat LS Diesel	2011	1.3	1.5	84,656	Diesel	Dealer	Manual	0
Maruti Suzuki Wagon R LXi Minor	2007	1.3	1.5	1,50,000	Petrol	Dealer	Manual	0
Maruti Suzuki SX4 ZXi	2009	1.4	1.6	58,000	Petrol	Dealer	Manual	0
Maruti Suzuki SX4 VXI CNG BS-IV	2010	1.51	1.71	76,000	CNG	Dealer	Manual	0
Ford Fiesta Exi 1.6 Duratec Ltd	2009	1.55	1.75	1,14,000	Petrol	Dealer	Manual	0
Hyundai i10 1.2 L Kappa Magna S	2009	1.6	1.8	60,000	Petrol	Dealer	Manual	0
Maruti Suzuki Wagon R LXi Minor	2009	1.6	1.8	49,000	Petrol	Dealer	Manual	0
Honda City ZX VTEC	2008	1.6	1.8	83,920	Petrol	Dealer	Manual	0
Maruti Suzuki Alto LXi BS-III	2008	1.6	1.8	67,181	Petrol	Dealer	Manual	0
Maruti Suzuki Alto K10 VXi	2012	1.61	1.81	63,250	Petrol	Dealer	Manual	0
Ford Fiesta Classic SXi 1.4 TDCi	2011	1.65	1.85	55,000	Diesel	Dealer	Automatic	0
Maruti Suzuki Alto K10 LXi	2012	1.7	1.9	60,028	Petrol	Dealer	Manual	0
Maruti Suzuki A-Star Vxi	2010	1.79	1.99	54,000	Petrol	Dealer	Manual	0
Maruti Suzuki Alto K10 VXi	2012	1.79	1.99	34,579	Petrol	Dealer	Manual	0
Maruti Suzuki Alto K10 VXi	2012	1.79	1.99	34,579	Petrol	Dealer	Manual	0
Tata Indigo eCS LS CR4 BS-IV	2014	1.7	2	1,75,000	Diesel	Dealer	Manual	0
Toyota Corolla H2 1.8E	2004	1.7	2	1,40,000	Petrol	Dealer	Manual	0
Maruti Suzuki Alto LXi BS-III	2009	1.7	2	60,506	Petrol	Dealer	Manual	0
Toyota Corolla H2 1.8E	2005	1.7	2	1,40,000	Petrol	Dealer	Manual	0
Maruti Suzuki Alto K10 LXi	2010	1.71	2.01	52,151	Petrol	Dealer	Manual	0

Level 2: Preprocessing

- Data Cleaning: Remove missing values, duplicates, and outliers from the dataset.
- Identify unique values: Examining the unique values in each feature of the dataset.
- Feature Engineering: Create new features from the existing variables.
- Dimensionality Reduction: Reduce the number of variables in the dataset by selecting only the required variables.



Level 2: Preprocessing

```
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
[0 1 3]
```

```
df.isnull().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

```
df.columns
```

```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

```
final_dataset = df[['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner']]
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

- We count the number of missing or null values in each column.
- In the code we selects specific columns from the dataframe and creates a new dataframe by removing the irrelevant columns from our dataset.

Level 2: Preprocessing

```
final_dataset['Current_Year'] = 2021
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021

```
final_dataset['Age'] = final_dataset['Current_Year']-final_dataset['Year']
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	Age
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

```
final_dataset.drop(['Current_Year'],axis=1,inplace=True)
```

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	7
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	8
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	4
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	10
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	7

```
print(df['Fuel_Type'].unique())
```

```
['Petrol' 'Diesel' 'CNG']
```

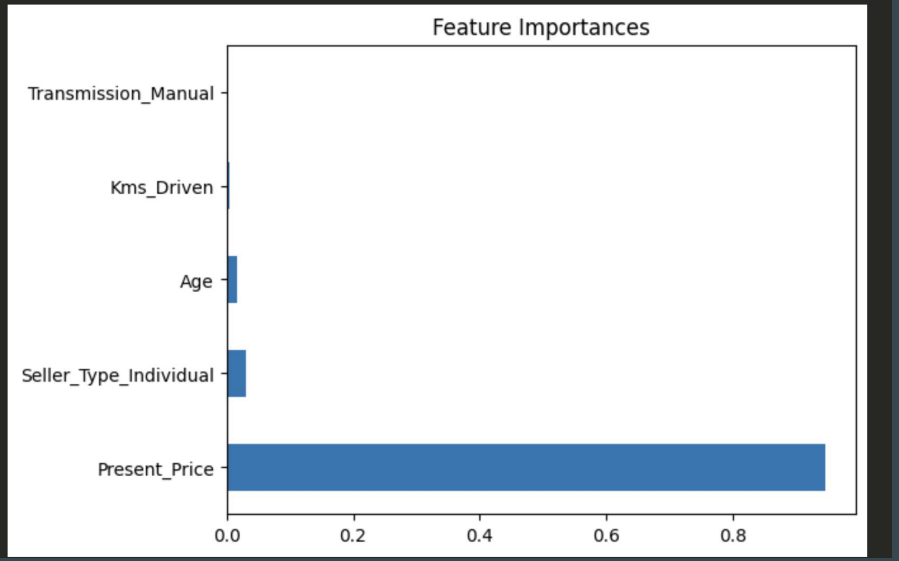
```
final_dataset=pd.get_dummies(final_dataset,drop_first=True)
```

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	7	0	1	0	1
1	4.75	9.54	43000	0	8	1	0	0	1
2	7.25	9.85	6900	0	4	0	1	0	1
3	2.85	4.15	5200	0	10	0	1	0	1
4	4.60	6.87	42450	0	7	1	0	0	1

Level 3: Modeling

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```



- We are for finding the feature importances using ExtraTreesRegressor
- By aggregating the results the algorithm estimates the relative importance of each feature in predicting the target variable.

Level 3: Modeling

Random Forest Regression

- Random Forest Regression is a machine learning algorithm that uses a collection of decision trees to make predictions.
- Each decision tree in the Random Forest is built using a random subset of the training data and a random subset of the input features, which helps to prevent overfitting.
- In this project, Random Forest Regression was used to predict the price of used cars based on attributes such as the car's age, mileage, fuel type, seller type, and more.
- The algorithm was trained using a dataset that included information on car names, years, selling prices, present prices, kilometers driven, fuel types, seller types, transmissions, and owners.
- Random Forest Regression was chosen for this project because it is a powerful algorithm that can handle both categorical and numerical data, and can detect complex nonlinear relationships between variables.

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
```

✓ 0.0s

```
import numpy as np
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]

print(n_estimators)
```

✓ 0.0s

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

```
#Randomized Search CV
max_features = ['auto', 'sqrt'] # we first consider all the featurees and
#then sqare root number of features to train the model

# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
#we create trees with 5 10 15 for each model...and train it

# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# we split as 2 nodes forst then 5 then 10 like that till 100 from the list

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

✓ 0.0s

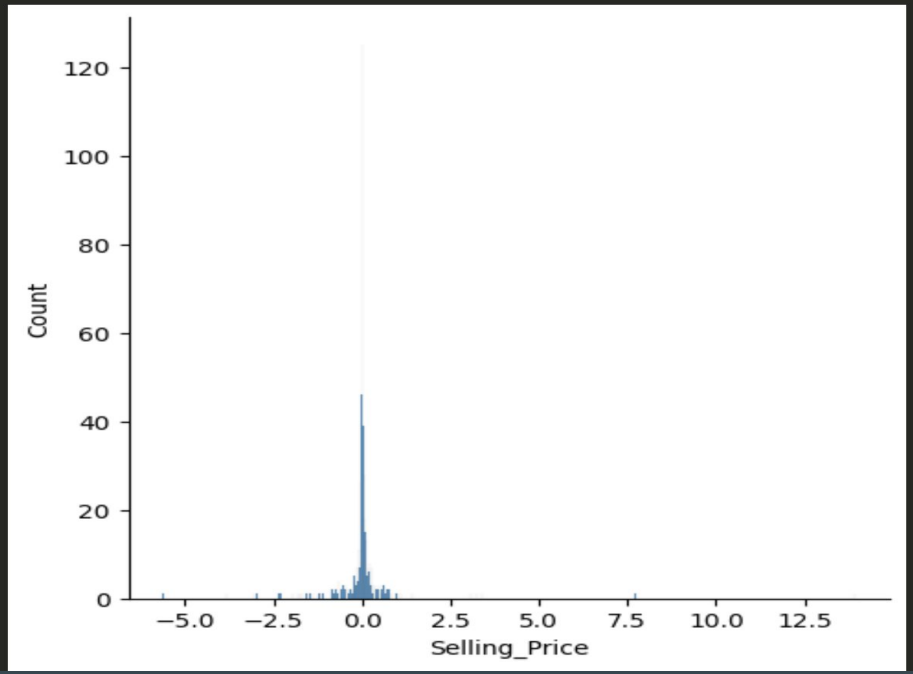
- Randomized Search CV was used in our project to optimize the hyperparameters of the Random Forest Regression model.
- The hyperparameters that were tuned included max_features, max_depth, min_samples_split, and min_samples_leaf.
- Two options were tried for max_features - considering all features and considering the square root of the number of features.
- For max_depth, six different values were tested - 5, 10, 15, 20, 25, and 30.
- Five different values were tested for min_samples_split - 2, 5, 10, 15, and 100.
- Four different values were tested for min_samples_leaf - 1, 2, 5, and 10.

Level 4: Analysis

```
sns.displot(y_test-predictions)
```

✓ 0.8s

```
<seaborn.axisgrid.FacetGrid at 0x2879a1c10>
```

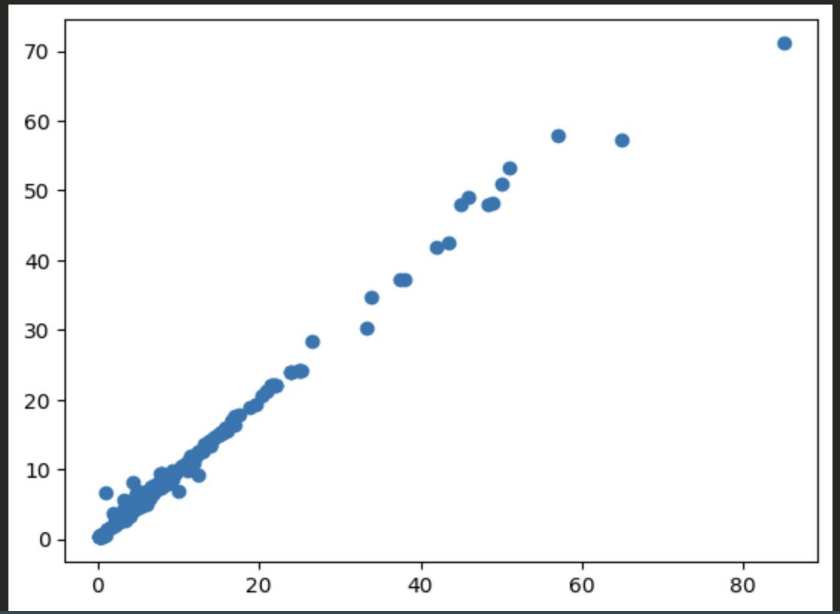


- sns.displot(y_test-predictions) creates a distribution plot of the residuals, i.e., the difference between the actual target values (y_test) and the predicted target values (predictions).
- It gives us an idea of how well the model is performing by showing the distribution of the errors.
- If the residuals are normally distributed around zero, it indicates that the model is performing well and the errors are random. However, if there is a pattern or skewness in the distribution, it indicates that the model is not performing well and there are systematic errors.
- In our case the selling price are normally distributed around 0, so the performance is good.

Level 4: Analysis

```
plt.scatter(y_test,predictions)
```

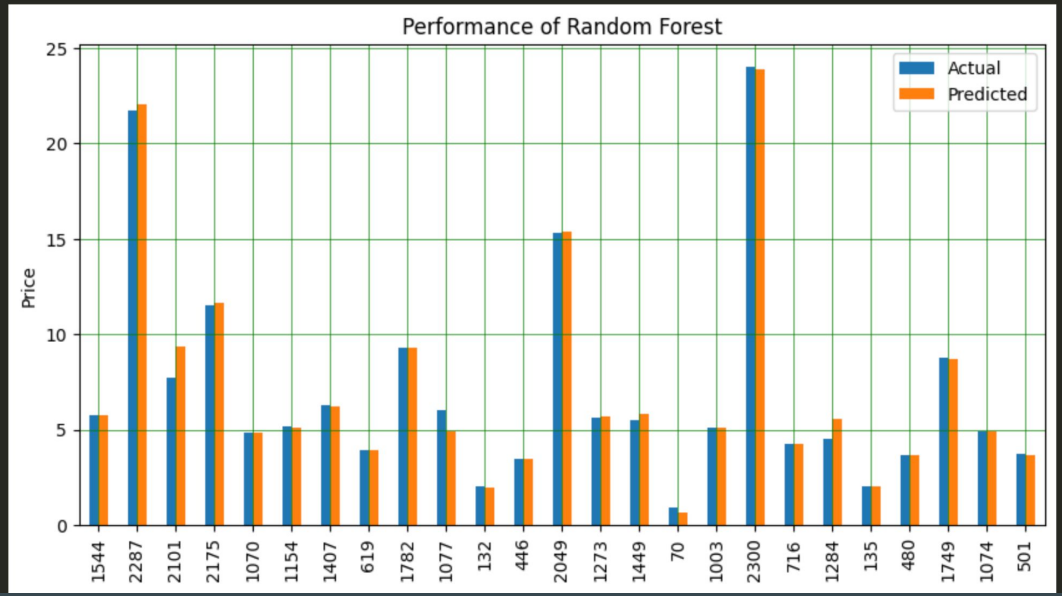
```
<matplotlib.collections.PathCollection at 0x2879a22d0>
```



- This code generates a scatter plot of the predicted values (along the y-axis) against the actual values (along the x-axis).
- The x-axis represents the actual values of the target variable (i.e., y_test), while the y-axis represents the predicted values.
- The scatter plot allows us to visualize how well the model's predictions match the actual values. In our case the points are close to a diagonal line, which indicates that the predicted values are very close to the actual values.

Level 4: Analysis

```
df_check = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
df_check = df_check.head(25)
df_check.plot(kind='bar',figsize=(10,5))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.title('Performance of Random Forest')
plt.ylabel('Price')
plt.show()
```

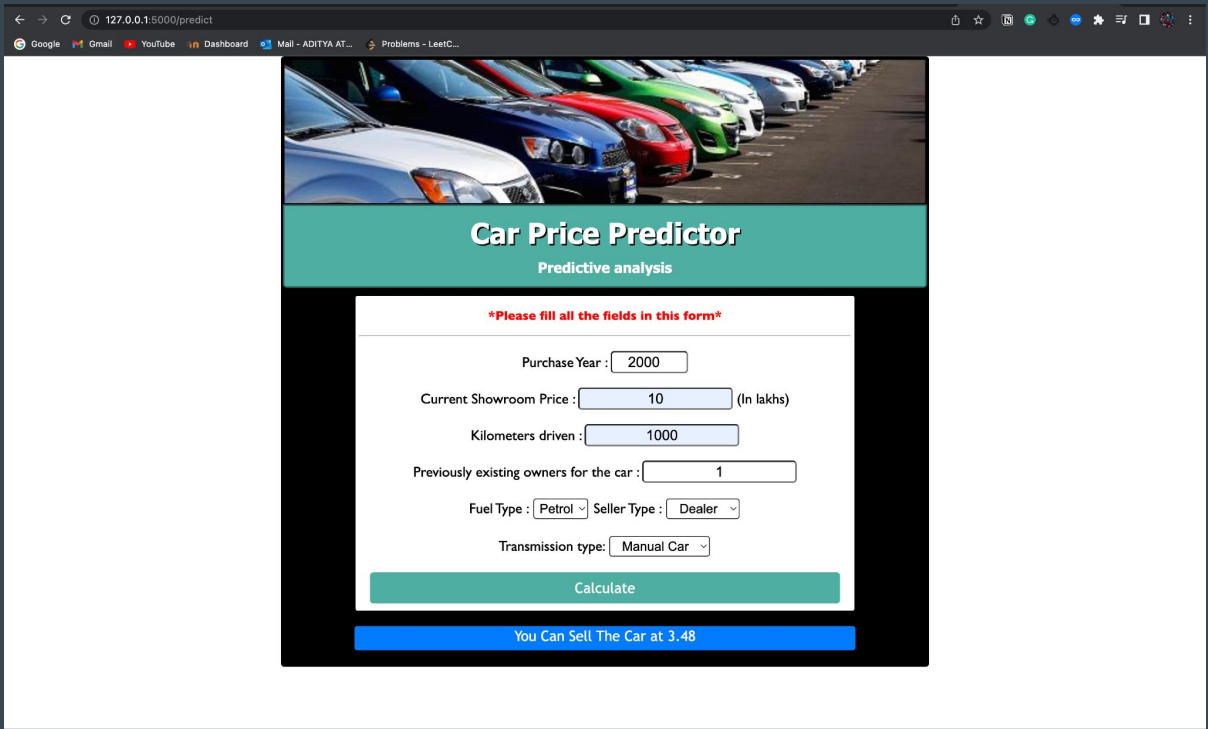


- Creates a DataFrame df_check with two columns 'Actual' and 'Predicted', where 'Actual' contains the actual values of y_test and 'Predicted' contains the predicted values using the trained model.
- This graph gives us the general idea about how the model is performing. In our case, the actual and predicted values are very close to each other indicating good accuracy.

Results

- We used two models for our project - Linear Regression and Random Forest Regression.
- Linear Regression gave an accuracy of around 90%, while Random Forest Regression gave an accuracy of around 95%.
- Although the difference between the two accuracies seems small, it can have a significant impact on the performance of the model.
- Random Forest Regression is a more complex model that can capture non-linear relationships and interactions between features.

Results



- The code uses Flask framework to create a web application.
- The trained Random Forest Regression model is loaded from a saved pickle file for prediction.
- We need the user to fill the details of his used car.
- Our website will use the model to predict the price of the car.