Phone Price Regression Model

Dr: Sara Suidan

Department: AI

➢ Team members:

- Idris Tarek Elsayed Ahmed Abdelaal
- Ahmed Abohadeed
- Rania Yasser Mohamed
- Khloud Elsayed Elsayed
- Eman Hamada Abdelhady

- Topics:
  - Abstract
  - Introduction
  - proposed work
  - commented code and some descriptions.

➢ Abstract:

In this study, we developed a machine learning regression model to predict the price of mobile phones based on their specifications. We used a dataset containing various features such as RAM, storage capacity, camera quality, and brand, among others, to train the model. The study aimed to investigate the effectiveness of machine learning regression models in predicting phone prices and identify the most significant features that influence phone prices. The results of this study can assist consumers and retailers in making informed decisions about purchasing and pricing mobile phones.

➢ Introduction:

- With the rapid advancement in technology and the ever-increasing demand for smartphones, mobile phone manufacturers are releasing new models with advanced features and capabilities at an unprecedented rate. As a result, consumers are faced with a wide range of options, making it challenging to choose the best phone that meets their needs and budget. Furthermore, pricing strategies for smartphones are complex, with several factors such as brand, features, and specifications influencing the price.
- To address this challenge, machine learning algorithms can be used to develop predictive models that can accurately estimate the price of mobile phones based on their specifications. Regression models are commonly used in machine learning for predicting continuous variables, making them suitable for predicting phone prices. By analyzing a large dataset of mobile phones with their corresponding prices, machine learning algorithms can identify patterns and relationships between phone features and their prices.

➢ proposed work:

1. **Data collection:** Collect a large dataset of mobile phones with their corresponding prices. The dataset should include various features such as RAM, storage capacity, camera quality, brand, etc.
2. **Data preprocessing:** Clean and preprocess the dataset by removing missing values, outliers, and irrelevant features. Normalize or standardize the data to ensure that all features are on the same scale.
3. **Feature selection:** Identify the most significant features that influence phone prices using feature selection techniques such as correlation analysis or feature importance ranking.
4. **Model selection:** Evaluate several regression algorithms such as linear regression, decision trees, and gradient boosting, among others. Select the algorithm that provides the best performance on the validation set.
5. **Model optimization:** Fine-tune the selected algorithm by adjusting hyperparameters and performing cross-validation to achieve the best possible performance on the test set.
6. **Model evaluation:** Evaluate the performance of the optimized model on the test set using metrics such as mean absolute error (MAE) and root mean squared error (RMSE).
7. **Deployment:** Deploy the model into a production environment, such as a mobile app or a web-based tool, to allow consumers and retailers to predict phone prices based on their specifications.

➢ commented code and some descriptions:

Imports the necessary libraries

#Importing Libraris

```
import numpy as np  #for matrix Value
import matplotlib.pyplot as plt  # Data Visulization
import seaborn as sns # Data Visulization
import pandas as pd  #For Reading flie .CSV
from sklearn.model_selection import train_test_split #Cross validation and split data for train and test
from sklearn.ensemble import RandomForestRegressor #Algorthim to build model
from sklearn.tree import DecisionTreeRegressor  #Algorthim to build model
from sklearn.neighbors import KNeighborsRegressor #Algorthim to build model
from sklearn.linear_model import LinearRegression  #
from xgboost import XGBRegressor
from sklearn import metrics # Find erorr function
```

[2] ✓ 0.3s Python

- This code imports data from a CSV file into a Pandas DataFrame, which can then be used for further data processing and analysis.

```
#Loading  data to Pandas Dataframe
path='Cellphone.csv'
phone_price=pd.read_csv(path)
#print the 5 first row
phone_price.head()
```

✓ 0.1s Python

- This code is checking for missing values (null values) in each column of the "phone_price" Pandas DataFrame using the "isnull()" function &&The ".sum()" method is then called on the Boolean DataFrame returned by "isnull()" to calculate the sum of null values in each column of the "phone_price" DataFrame. The output of this code is a Series that shows the total number of null values in each column of the "phone_price" DataFrame.

```
#Finding if there data is null
phone_price.isnull().sum()
```

✓ 0.0s Python

```
Product_id      0
Price           0
Sale            0
```

- This code generates a statistical summary of the "phone_price" Pandas DataFrame. The "describe()" function calculates various summary statistics for each column in the DataFrame,

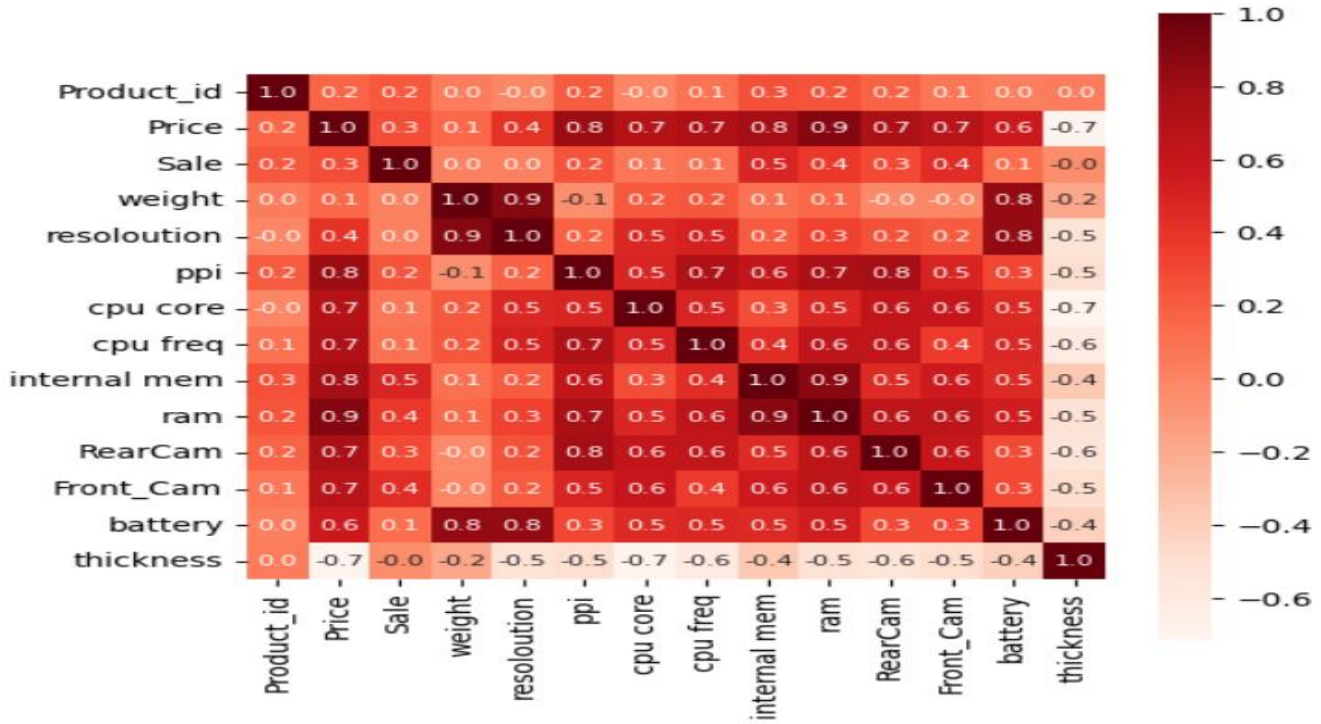
- This code calculates the correlation matrix of the "phone_price" Pandas DataFrame using the "corr()" function.
- The output of this code assigns the resulting correlation matrix to a new variable called "correlation". This variable can be used to analyze the relationships between variables in the dataset. For example, positive correlations (values close to 1) indicate that two variables tend to increase or decrease together, while negative correlations (values close to -1) indicate that two variables tend to move in opposite directions.

- The code you provided is used to create a heatmap plot of a correlation matrix using the Python libraries matplotlib and seaborn.

```python
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Reds')
plt.show()
```

✓ 1.7s Python

➢ The output :

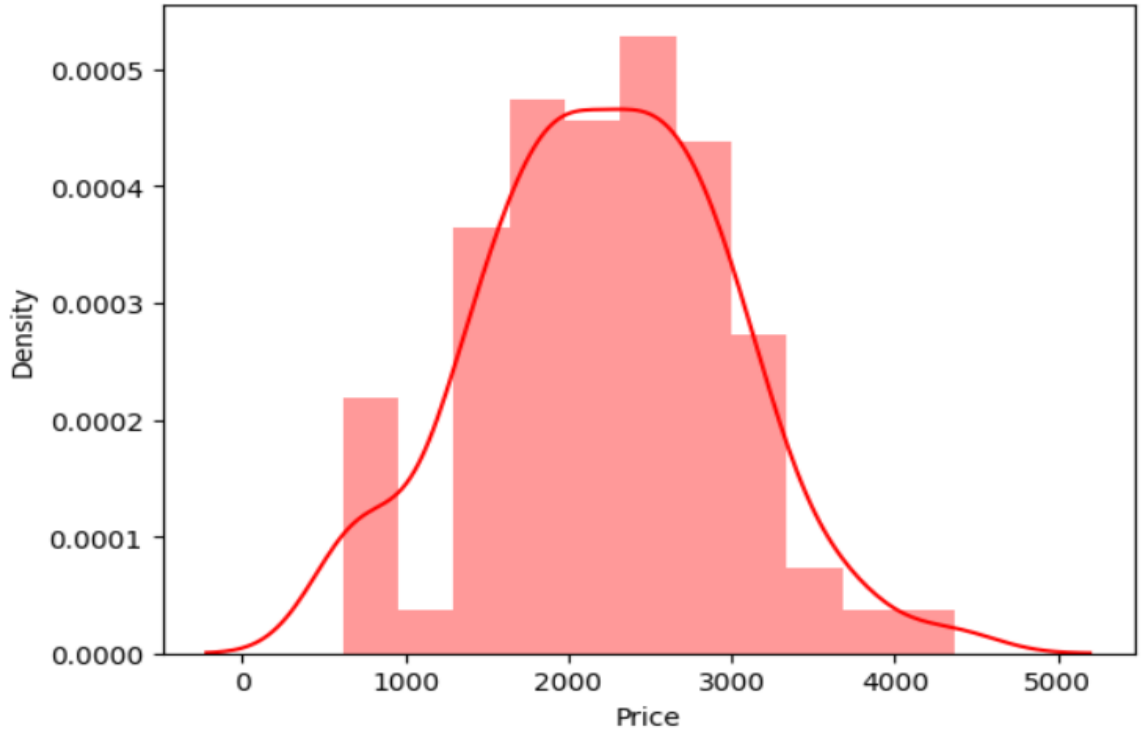


- sns.distplot() is a function that creates a histogram and a kernel density estimate plot of a variable. The phone_price['Price'] argument specifies the variable to be plotted, which is the 'Price' column of the DataFrame phone_price.

```
# checking the distribution of the phone Price
sns.distplot(phone_price['Price'],color='red')
```

✓ 0.6s Python

➢ The output :



- This code is an example of how to split a dataset into training and testing sets using the train_test_split() function from the scikit-learn library in Python.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)
```
✓ 0.1s Python

➢ Model Training

1. Random Forest Regression model

```
model=RandomForestRegressor(n_estimators=120)
model.fit(X_train,Y_train)
```
✓ 0.4s Python

➢ Model Evalution:

```
predict_test=model.predict(X_test)
print(predict_test)
```
✓ 0.0s

```
model.score(X_train,Y_train)
```
✓ 0.1s

```
# R squared error
error_score = metrics.r2_score(Y_test, predict_test)
print("R squared error : ", error_score)
```
✓ 0.1s

2. Linear Regession Model

```
model=LinearRegression()
model.fit(X_train,Y_train)
```

✓ 0.0s Python

3. Decision Tree Regressor Model

```
model=DecisionTreeRegressor(max_depth=10)
model.fit(X_train,Y_train)
```

✓ 0.0s Python

4. Kneighbors Regressor Model

```
model=KNeighborsRegressor()
model.fit(X_train,Y_train)
```

✓ 0.0s Python

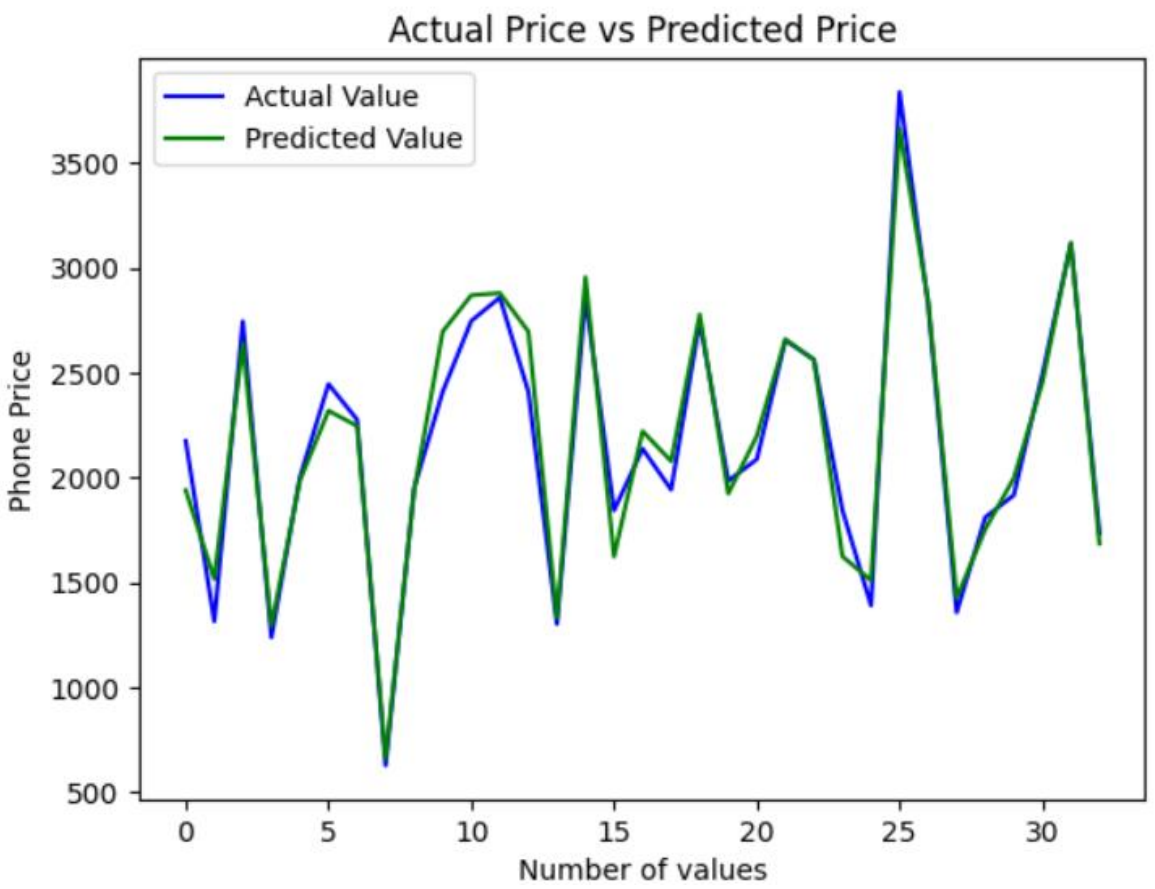
5. XGB Regressor Model

```
model=XGBRegressor()
model.fit(X_train,Y_train)
```

✓ 1.2s Python

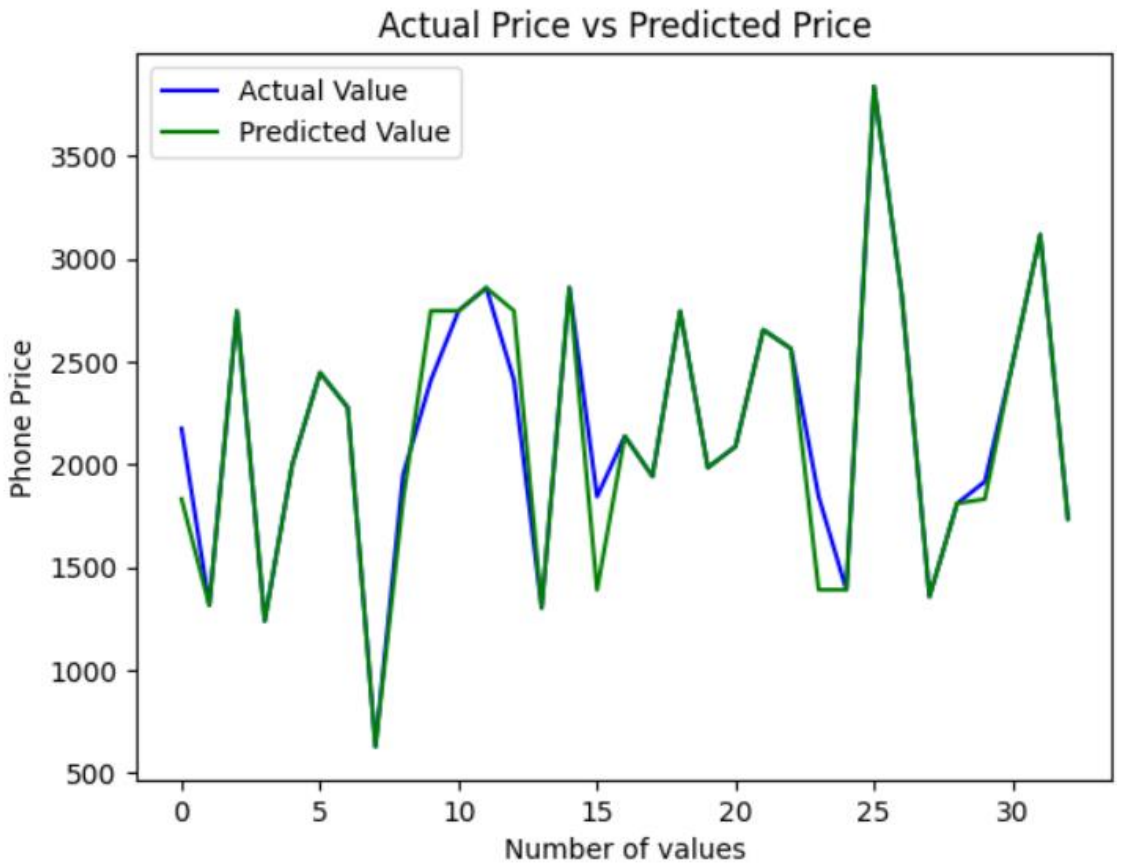
- Compare the Actual Values and Predicted Values in a Plot

In Random Forest Regression model



In Linear Regession model

In Decision Tree Regressor model



In Kneighbors Regressor model

In XGB Regressor model