Project Title : Cars Price Prediction

Date : 16-06-2023 Prepared By : SURIYAKRISHNAN

Project Justification :
In This project to predict the cars Price its depends on the cars features, and this is a Regression datasets(Regression problem).

Project Requirements :

✓ Dataset

✓ Python's libraries(necessory)

Import the necessory libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from scipy import stats
import time
from sklearn.feature_selection import VarianceThreshold
from imblearn.over_sampling import SMOTE, RandomOverSampler
from xgboost import XGBRegressor
```

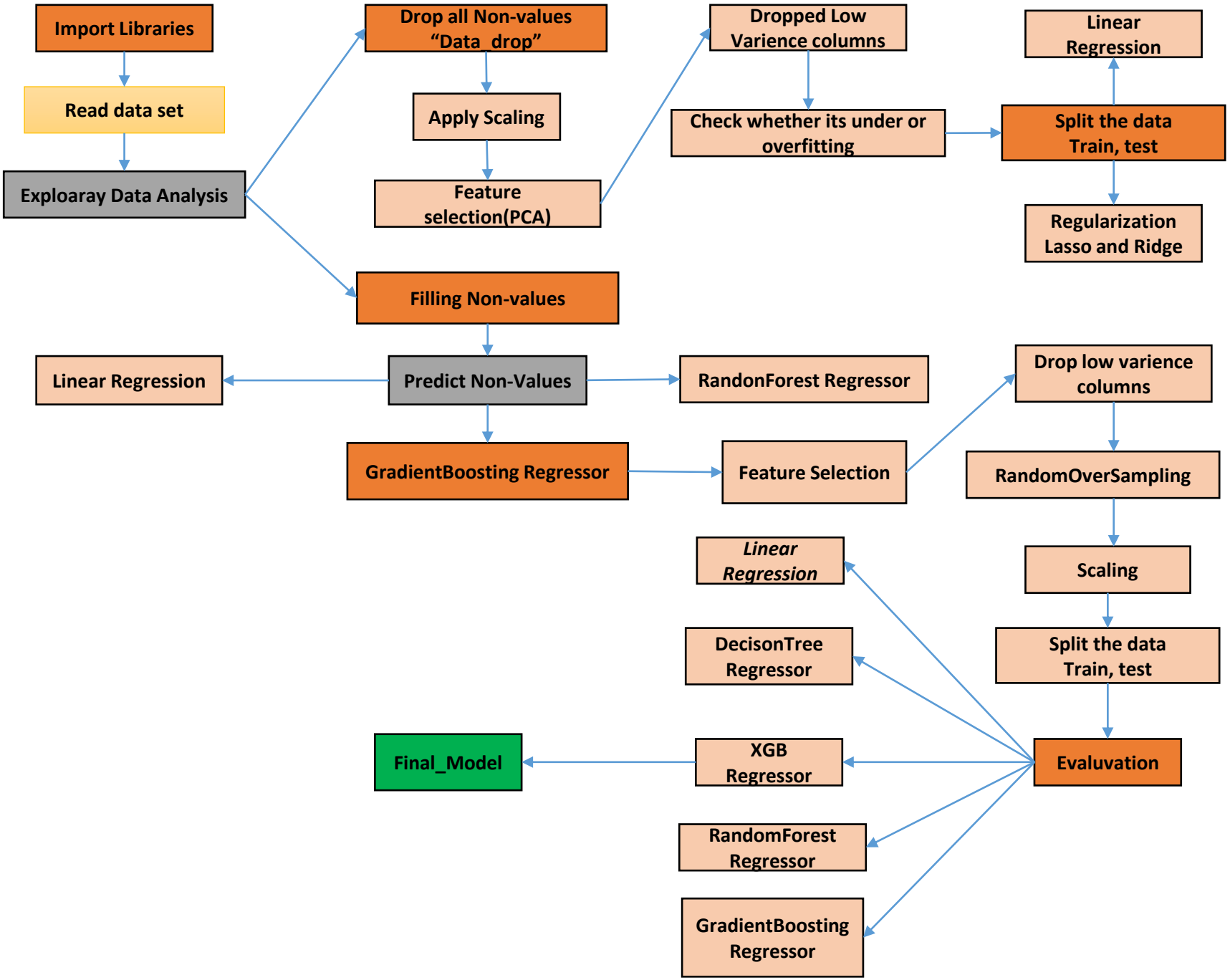
First need to import or load the necessary and related libraries, Some very primary and almost necessary packages for Machine Learning are — NumPy, Pandas, Matplotlib and Scikit-Learn.

Dataset details:

- ✓ Dataset file: 'cars_price.csv'
- ✓ This is a regression data set.
- ✓ The data set has 206 samples.
- ✓ There are 25 features.
- ✓ The target variable is the price of the car.

Description of attributes:

- ✓ symboling
- ✓ normalized-losses
- ✓ make
- ✓ fuel-type
- ✓ aspiration
- ✓ num-of-doors
- ✓ body-style
- ✓ drive-wheels
- ✓ engine-location
- ✓ wheel-base
- ✓ length
- ✓ width
- ✓ height
- ✓ curb-weight
- ✓ engine-type
- ✓ num-of-cylinders
- ✓ engine-size
- ✓ fuel-system
- ✓ bore
- ✓ stroke
- ✓ compression-ratio
- ✓ horsepower
- ✓ peak-rpm
- ✓ city-mpg
- ✓ highway-mpg


Import Libraries
Read data set
Exploaray Data Analysis
Drop all Non-values "Data_drop"
Apply Scaling
Feature selection(PCA)
Dropped Low Varience columns
Check whether its under or overfitting
Linear Regression
Split the data Train, test
Regularization Lasso and Ridge
Filling Non-values
Linear Regression
Predict Non-Values
RandonForest Regressor
GradientBoosting Regressor
Feature Selection
Drop low varience columns
RandomOverSampling
Scaling
Split the data Train, test
Evaluvation
Linear Regression
DecisonTree Regressor
XGB Regressor
RandomForest Regressor
GradientBoosting Regressor
Final_Model

```
# read the dataset
data= pd.read_csv('cars_price.csv')
```

Once the libraries are loaded, need to get the data loaded. Pandas has a very straightforward function to perform this task — pandas.read_csv. The read.csv function is not just limited to csv files, but also can read other text based files as well.

Explorary Data Analysis:

- ✓ To check how many duplicates values in the dataset
- ✓ This dataset have a many unvalid values in the features{?}. All the unvallid entries go replace np.Non-values.

```
data= data.replace('?', np.nan)
```

- ✓ This dataset have a many unvalid values in the features{?}. All the unvallid entries go replace np.Non-values.
- ✓ To check how many unique values are there in each columns. And apply Encoading to object coumns in dataset

Data_drop:

- ✓ Drop all Non-values in the dataset, this data is called by 'data_drop' and this is carried seperatly for how this dataset is working , how many scores produce

```
# change object into float
cols= ['bore','stroke']
for col in cols:
  data_drop[col]= data_drop[col].astype(float)
```

- ✓ Some columns are values given by numerical but that columns is object class, that's why that columns is change object into float or integer
- ✓ Split the dataset for X and y, X have a independent coumns and y have a one dependent coumn('price')
- ✓ Visulize the most perfect fit line using slop,intercept and standard_error to scatter plot

- ✓ Inside the scatter plot x='make' and y='price'

Feature selction for drop_data:

- ✓ Impleament PCA and standardscaler to check how many low varience columns are there and some coumns are dropped.

```
train_err=[]
test_err=[]
for index in range(100):
  X_train,X_test,y_train,y_test= train_test_split(drop_x_train,drop_y_train, test_size=0.2)
  ss= StandardScaler()
  X_train=ss.fit_transform(X_train)
  X_test=ss.transform(X_test)
  lr.fit(X_train, y_train)
  predicted_y_train= lr.predict(X_train)
  predicted_y_test= lr.predict(X_test)
  train_err.append(mean_squared_error(y_train,predicted_y_train))
  test_err.append(mean_squared_error(y_test,predicted_y_test))

print('mean_squared_error train', sum(train_err)/len(train_err))
print('mean_squared_error test', sum(test_err)/len(test_err))
```

- ✓ Using for loop to split the Train and Test for 100 times and fit Linear Regression , and check the average error in Train and Test for Whether this dataset is overfitting or underfitting.
- ✓ The errors are nearly by Train and Test so far this dataset balanced.

Evaluation for data_drop:

Linear Regression giving r2_score= 0.81 and need to improve model r2 score

Regularization:

L2 regularization giving r2_score= 0.75

Data Preprocessing for the Original dataset:

```
data['bore']= data['bore'].fillna(method='pad')
data['stroke']= data['stroke'].fillna(method='pad')
data['horsepower']= data['horsepower'].fillna(method='pad')
data['peak-rpm']= data['peak-rpm'].fillna(method='pad')
data['price']= data['price'].fillna(method='pad')
```

- ✓ In the dataset 'normalized-losses' have a many Non-values compared to other columns. So that predict Non-values
- ✓ Beforly filled some other columns Non-values using 'pad' method because if any Non-values in independent columns its will give error
- ✓ And now split the dataset for predict 'normalized-losses' Non-values , X is independent features and y is 'normalized-losses'
- ✓ Using for loop to fit some most commonly using moduls(Linear Regression, GradientBoosting Regressor and RandomForest Regressor) and calculate which model is giving best r2_score
- ✓ In this case GradientBoosting Regressor give best r2_score compared to Linear and Randomforest

```
gbr= GradientBoostingRegressor()
gbr.fit(X,y)
fill= gbr.predict(to_predict)
to_predict['normalized-losses'] = fill  #filling the null values
data = pd.concat([data.dropna(),to_predict]).reindex(data.index)  #adding the rows in which total_bedrooms was null an
```

- ✓ Fill the Non-values appling GradientBoosting Technique to predict the Non-vaues in the normalized-losses
- ✓ And merge that predicted_data to Original dataset using concat method

Feature selection:

PCA and scaling applied in dataset and to show barplot to how low varience columns are there in the dataset. Low varience columns are dropped from dataset

Oversampling:

- ✓ RandomOversampler used to Oversampling the dataset
- ✓ And split the dataset for Train and Test (Trained data have a 80% of the data and Test have 20% of the data)
- ✓ Apply Standardscaler to X_train and X_test

Evaluvation:

Model	mean_absolute_error	mean_squared_error
✓ *Linear Regression*	*: 2030.45*	*7213041.43*
✓ *XGB Regressor*	*: 92.31*	*76761.00*
✓ **DecisionTree Regressor**	**: 56.01**	**50097.51**
✓ *RandomForest Regressor*	*: 380.05*	*297644.92*
✓ *GradientBoosting Regressor*	*: 595.11*	*660589.83*

Model	r2_score
✓ *Linear Regression*	*: 0.8639*
✓ *XGB Regressor*	*: 0.9985*
✓ **DecisionTree Regressor**	**: 0.9990**
✓ *RandomForest Regressor*	*: 0.9943*
✓ *GradientBoosting Regressor*	*: 0.9875*

Final_Model:

- ✓ DecisionTree Regressor and XGB regressor have good r2_score so I finalize the model to DecisonTree Regressor, because DecisionTree Regressor produce good r2_score and minimum mean_squared_error better than XGB Regressor

Final_Model	r2_score	mean_absolute_error	mean_squared_error
DecisionTree Regressor	**0.9990**	**57.98**	**51516.25**