

Machine learning

Milestone1 - Report

Data Preprocessing

1. preprocessing techniques

a. Missing values:

Dataset contained Some missing values so we dropped any row that contained at least one missing value, but we have done this after choosing the features that we are going to use.

```
data.dropna(how='any',inplace=True)
```

b. Text values:

Since computers can't deal with string values, so we need to convert any text data to numbers, so we have changed Prime_genre column to numeric values using One-Hot-Encoding technique.

```
one_hot = pd.get_dummies(data['prime_genre'])  
data = data.join(one_hot)  
data = data.drop(['prime_genre'],axis=1)
```

c. Feature scaling:

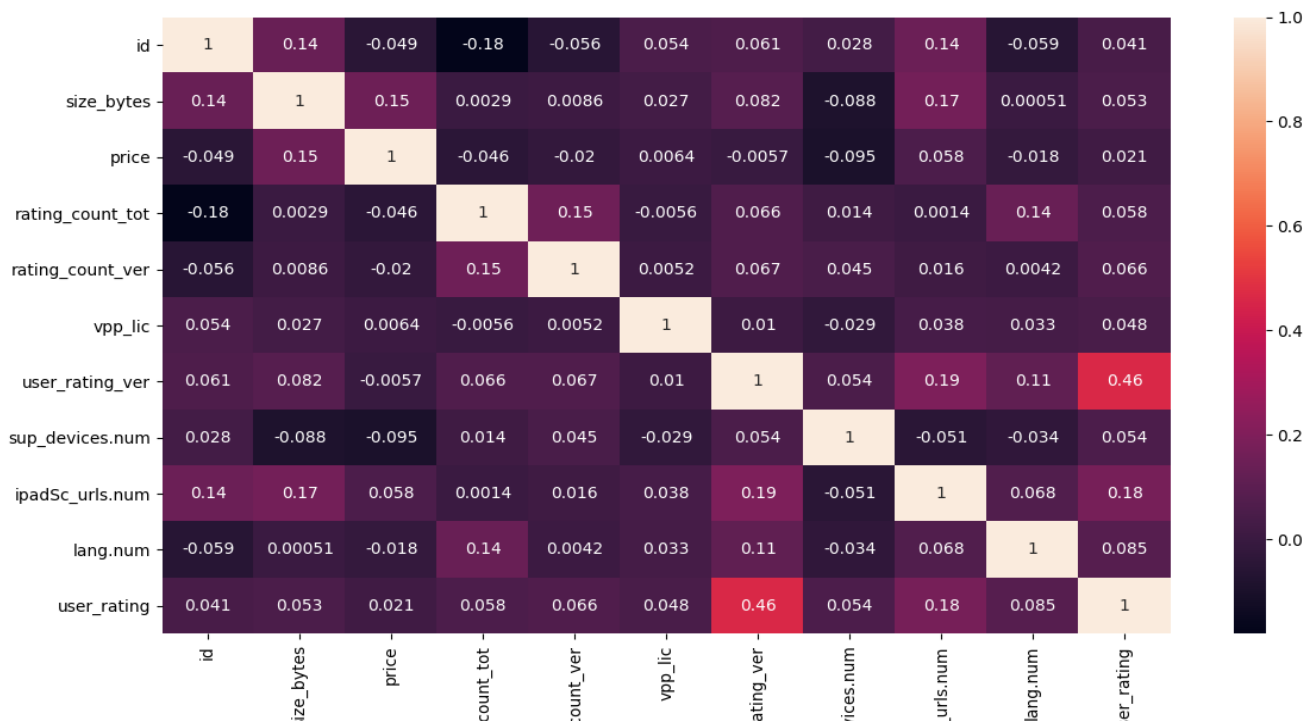
Features values differ very much in ranges between them so some attributes would affect the output more than other attributes only because it has higher ranges, so it is important to apply feature scaling. We applied min-max scaling on two columns ['rating_count_tot','rating_count_ver']

```
scaler = MinMaxScaler()  
data['rating_count_tot'] =  
scaler.fit_transform(np.array(data['rating_count_tot'])  
.reshape(-1,1))
```

```
data['rating_count_ver'] =
scaler.fit_transform(np.array(data['rating_count_ver'])
.reshape(-1,1))
```

2. Analysis

- Applied correlation between features to see which features have the most effect on the output (User_Rating) and decide which features we are going to use in our regression models



- After Applying correlation it was observed that the user_rating_ver has the most effect on the target value

3. Features Used

Feature	Description
rating_count_tot	User Rating counts (for all version)
rating_count_ver	User Rating counts (for current version)
user_rating_ver	Average User Rating value (for all version)
prime_genre	Primary Genre
sup_devices.num	Number of supporting devices
lang.num	Number of supported languages
ipadSc_urls.num	Number of screenshots showed for display

Regression Techniques

As we know that regression models are used to describe relationships between variables by fitting a line to the observed data, Regression allows you to estimate how a dependent variable changes as the independent variable changes.

So ,we used two techniques, Multiple linear regression and Polynomial Regression.

We also tried **enhancing** the results using **L2** regularization.

- **Multiple Linear Regression:**

$$Y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$$

Multiple Linear regression are based on the assumption that there is a linear relationship between both the dependent and independent variables or predictor variable and target variable, and it also assumes that there's no major correlation between the independent variables.

Multi linear regression can be linear and nonlinear, and it has one y and one or more x.

After Implementation of the multiple linear regression

We got a mean square error between 0.37652360484071495 and 0.35761225751504806

The root of mean square error is between

0.6136151928046721 and 0.5963103640175121

The R2 score is between

0.22929096545166372 and 0.21844404907891346

The screenshot shows a Python IDE with three files open: `dataProcessing.py`, `Multiple Regression.py`, and `PolynomialRegression.py`. The `Multiple Regression.py` file is active, showing the following code:

```
53 ridge_coefficient['Coefficient Estimate'] = pd.Series(ridgeR.coef_)
54 print(ridge_coefficient)
55
56 fig, ax = plt.subplots(figsize=(20, 10))
57
58 color=['tab:gray', 'tab:blue', 'tab:orange',
59 'tab:green', 'tab:red', 'tab:purple', 'tab:brown',
60 'tab:pink', 'tab:gray', 'tab:olive', 'tab:cyan',
61 'tab:orange', 'tab:green', 'tab:blue', 'tab:olive']
62
63 ax.bar(ridge_coefficient["Columns"],
64 ridge_coefficient['Coefficient Estimate'],
```

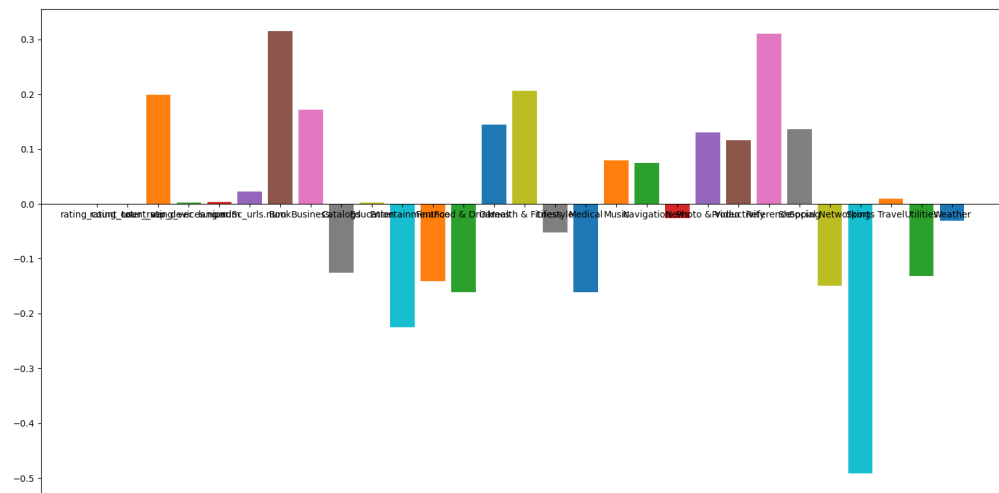
The terminal output shows the results of the Ridge Regression:

```
*****Multiple Linear Regression*****
Co-efficient of linear regression [ 1.16371247e-07  4.57867414e-06  2.05184846e-01 -4.51060089e-04
 2.72117821e-03  1.75579380e-02  2.72812845e-01  1.14196451e-01
 2.12635904e-01 -2.60577048e-02 -2.23628784e-01 -1.65170894e-01
 -9.29678307e-02  1.23989516e-01  2.05442205e-01 -2.54483121e-01
 -2.74100250e-01  6.85299503e-02  1.20513197e-01 -1.33468758e-01
 1.19473039e-01  1.26327900e-01  2.97056576e-01  1.90226019e-01
 -1.34403692e-01 -3.69816004e-01  6.81388830e-02 -1.80477926e-01
 -6.47675198e-02]
Intercept of linear regression model 3.179913188851388
Mean Square Error 0.36761225751504806
True App rate is : 4.5
Predicted App rate is : 4.414377899551721
True value for the first app in the test set is : 4.5
Predicted value for the first app in the test set is : 4.414377899551721
RMSE : 0.6063103640175121
Coefficient of determination : 0.22929096545166372
```

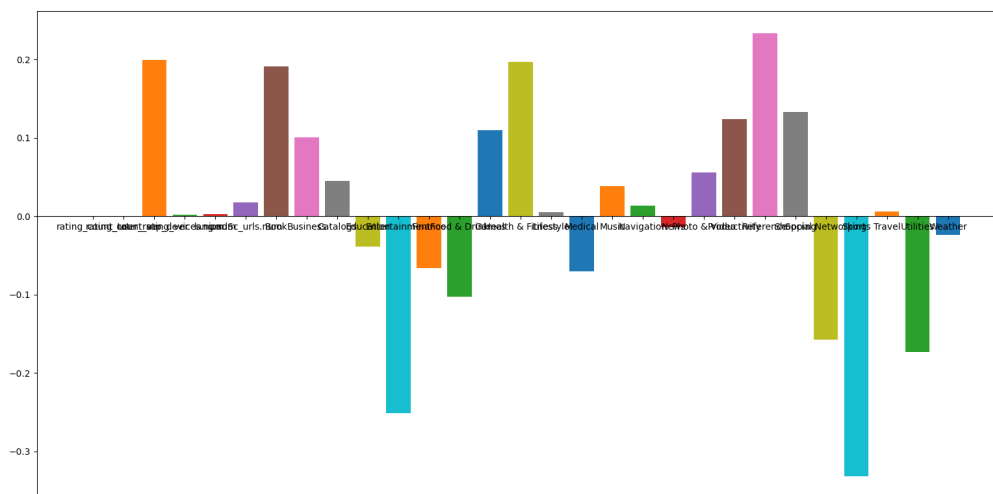
- **L2 regularization**

Ridge Regression added a term in ordinary least square error function that regularizes the value of coefficients of variables. This term is the sum of squares of coefficient multiplied by the parameter. The motive of adding this term is to penalize the variable corresponding to that coefficient not very much correlated to the target variable. This term is called L2 regularization.

alpha = 1



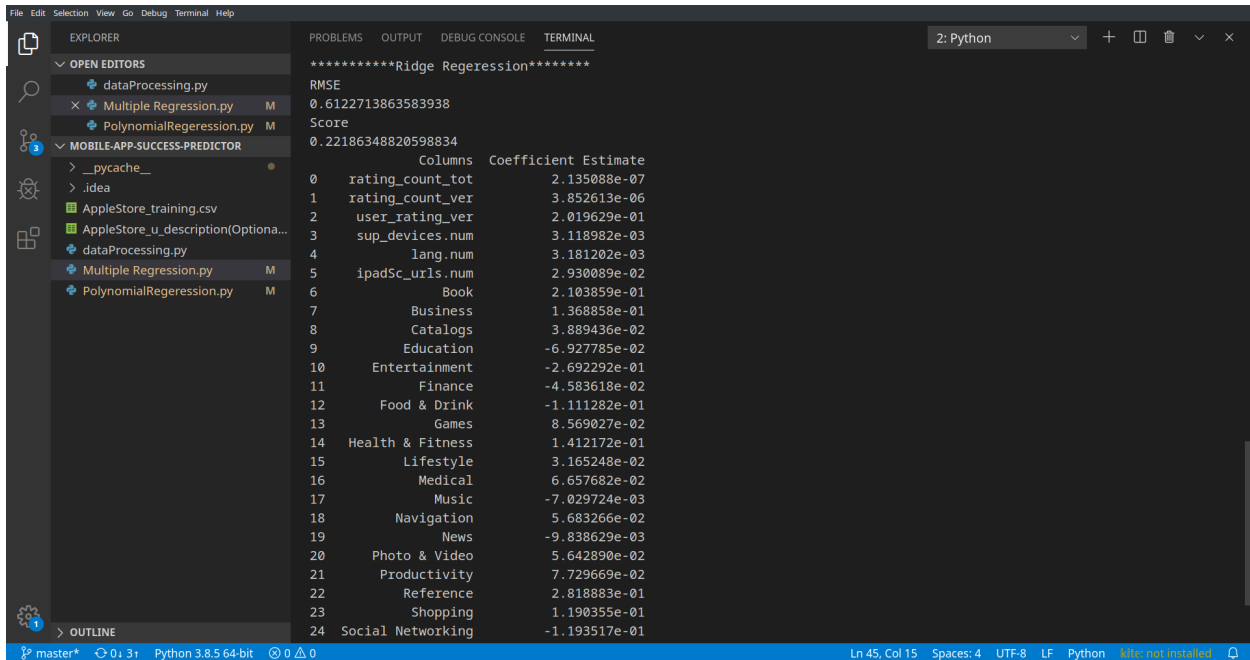
alpha = 10



As we can observe from the above plots that alpha helps in regularizing the coefficient and make them converge faster.

The score value is increased by (0.002 to 0.05)

And the error value is decreased by (0.01 to 0.03)



```
*****Ridge Regression*****
RMSE
0.6122713863583938
Score
0.22186348820598834

Columns      Coefficient Estimate
0    rating_count_tot    2.135088e-07
1    rating_count_ver    3.852613e-06
2    user_rating_ver      2.019629e-01
3    sup_devices.num      3.118982e-03
4    lang.num             3.181202e-03
5    ipadSc_urls.num      2.930089e-02
6    Book                 2.103859e-01
7    Business             1.368858e-01
8    Catalogs             3.889436e-02
9    Education            -6.927785e-02
10   Entertainment       -2.692292e-01
11   Finance             -4.583618e-02
12   Food & Drink        -1.111282e-01
13   Games               8.569027e-02
14   Health & Fitness     1.412172e-01
15   Lifestyle           3.165248e-02
16   Medical             6.657682e-02
17   Music               -7.029724e-03
18   Navigation          5.683266e-02
19   News                -9.838629e-03
20   Photo & Video       5.642890e-02
21   Productivity        7.729669e-02
22   Reference           2.818883e-01
23   Shopping            1.190355e-01
24   Social Networking   -1.193517e-01
```

• Polynomial Regression

$$Y = b_0 + b_1X + b_2X^2 + \dots + b_nX^n$$

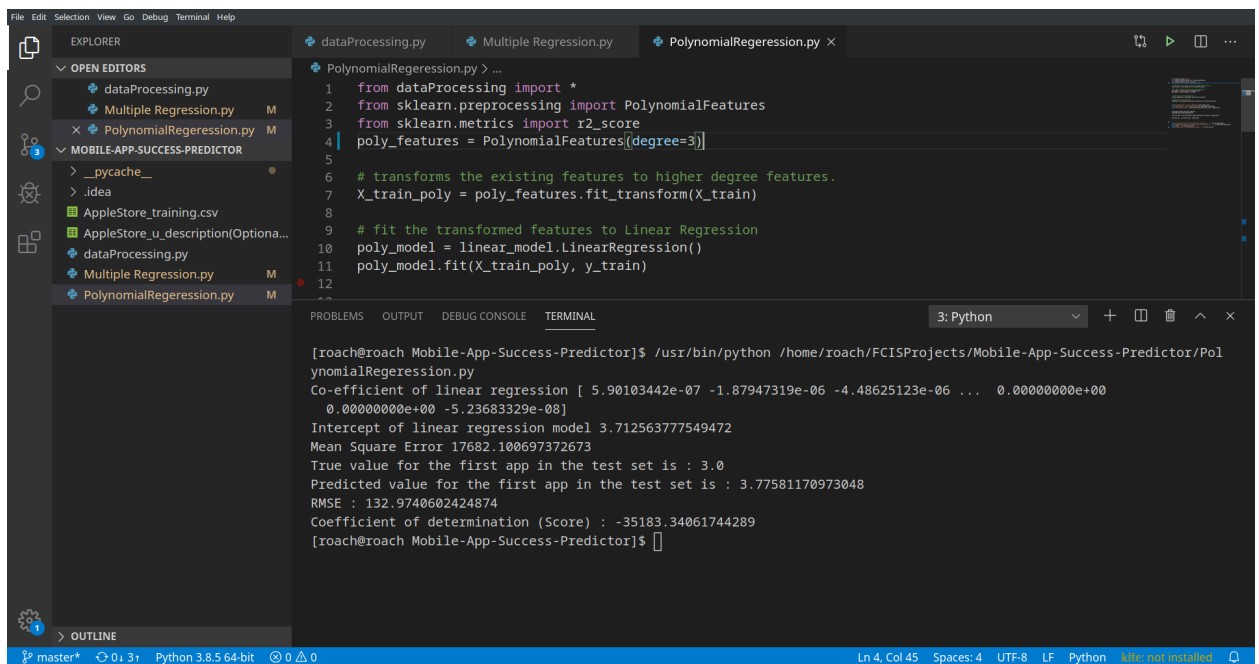
Polynomial regression is one of the types of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as n th degree polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted $E(y|x)$.

Polynomial regression provides the best approximation of the relationship between the dependent and independent variable.

→ First : Degree of 3

Produced very high root mean square error

Resulting in overfitting.



The screenshot shows an IDE with three open files: `dataProcessing.py`, `Multiple Regression.py`, and `PolynomialRegeression.py`. The `PolynomialRegeression.py` file is active, showing the following code:

```
1 from dataProcessing import *
2 from sklearn.preprocessing import PolynomialFeatures
3 from sklearn.metrics import r2_score
4 poly_features = PolynomialFeatures(degree=3)
5
6 # transforms the existing features to higher degree features.
7 X_train_poly = poly_features.fit_transform(X_train)
8
9 # fit the transformed features to Linear Regression
10 poly_model = linear_model.LinearRegression()
11 poly_model.fit(X_train_poly, y_train)
12
```

The terminal output shows the results of running the script:

```
[roach@roach Mobile-App-Success-Predictor]$ /usr/bin/python /home/roach/FCISProjects/Mobile-App-Success-Predictor/PolynomialRegeression.py
Co-efficient of linear regression [ 5.90103442e-07 -1.87947319e-06 -4.48625123e-06 ... 0.00000000e+00
0.00000000e+00 -5.23683329e-08]
Intercept of linear regression model 3.712563777549472
Mean Square Error 17682.100697372673
True value for the first app in the test set is : 3.0
Predicted value for the first app in the test set is : 3.77581170973048
RMSE : 132.9740602424874
Coefficient of determination (Score) : -35183.34061744289
[roach@roach Mobile-App-Success-Predictor]$
```

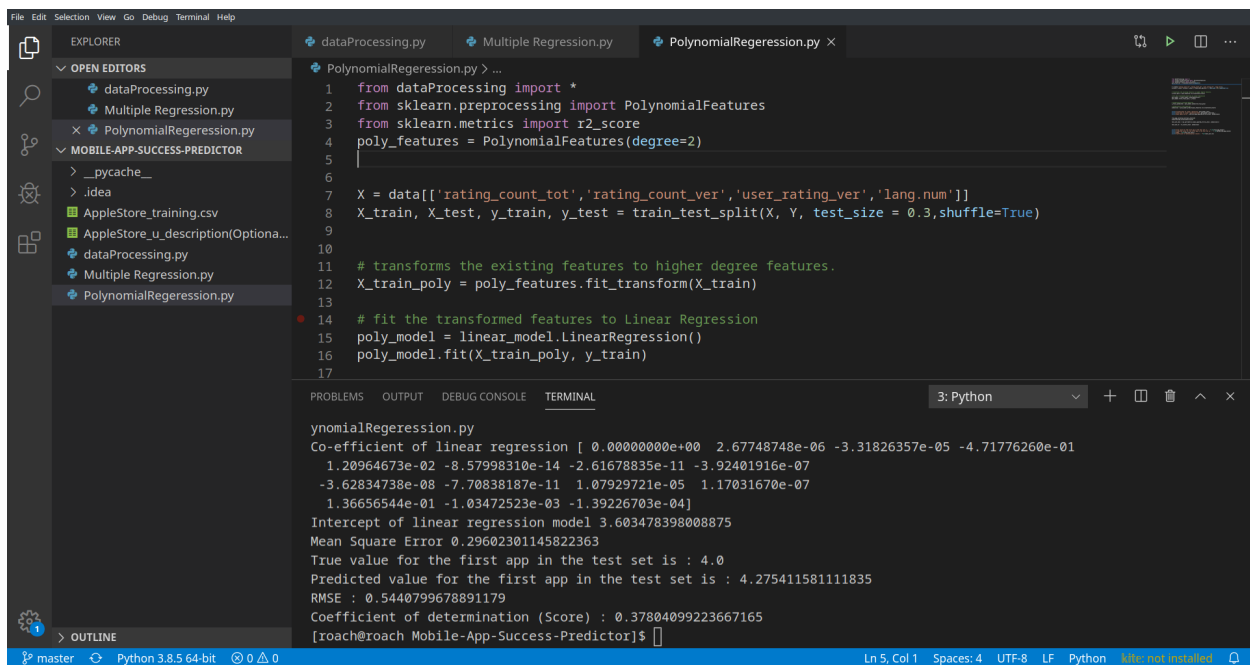

→ Second : Degree of 2

Produced lower root mean square error and higher R2 score than multiple linear regression.

Providing the better solution with this data.

Features used :

[rating_count_tot, rating_count_ver, user_rating_ver, lang.num]



The screenshot shows a VS Code editor with three files open: `dataProcessing.py`, `Multiple Regression.py`, and `PolynomialRegression.py`. The `PolynomialRegression.py` file is active, showing the following code:

```
1 from dataProcessing import *
2 from sklearn.preprocessing import PolynomialFeatures
3 from sklearn.metrics import r2_score
4 poly_features = PolynomialFeatures(degree=2)
5
6
7 X = data[['rating_count_tot', 'rating_count_ver', 'user_rating_ver', 'lang.num']]
8 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, shuffle=True)
9
10
11 # transforms the existing features to higher degree features.
12 X_train_poly = poly_features.fit_transform(X_train)
13
14 # fit the transformed features to Linear Regression
15 poly_model = linear_model.LinearRegression()
16 poly_model.fit(X_train_poly, y_train)
17
```

The terminal output shows the results of the polynomial regression model:

```
ynomialRegeression.py
Co-efficient of linear regression [ 0.00000000e+00  2.67748748e-06 -3.31826357e-05 -4.71776260e-01
 1.20964673e-02 -8.57998310e-14 -2.61678835e-11 -3.92401916e-07
 -3.62834738e-08 -7.70838187e-11  1.07929721e-05  1.17031670e-07
 1.36656544e-01 -1.03472523e-03 -1.39226703e-04]
Intercept of linear regression model 3.603478398008875
Mean Square Error 0.29602301145822363
True value for the first app in the test set is : 4.0
Predicted value for the first app in the test set is : 4.275411581111835
RMSE : 0.5440799678891179
Coefficient of determination (Score) : 0.37804099223667165
[roach@roach Mobile-App-Success-Predictor]$
```

Time of training

The time of training was between 7-10 seconds

Except the polynomial of degree 3 which took around 2-3 minutes.

Conclusion

- Time of training increases with the increase of the complexity.
- In higher polynomial degrees overfitting can occur.
- L2 regularization increases the accuracy of training.
- Data preprocessing and good feature selection leads to better results with better accuracy.

Machine learning

Milestone2 - Report

Data Preprocessing

1. preprocessing techniques

a. Missing values:

Dataset contained Some missing values so we dropped any row that contained at least one missing value, but we have done

```
#drop null rows  
data.dropna(how='any',inplace=True)
```

this after choosing the features that we are going to use.

b. Text values:

Since ML Algorithms can't read text value data, we need to convert any text data to numbers, so we have changed Prime_genre column to numeric values using One-Hot-Encoding technique.

```
#Onehot encoding X values  
dummy=pd.get_dummies(Classify_X['prime_genre'],prefix="Genre",drop_first=False)  
Classify_X=pd.concat([Classify_X,dummy],axis=1)  
Classify_X=Classify_X.drop(['prime_genre'],axis=1)
```

c. Feature scaling:

Features values differ very much in ranges between them so some attributes would affect the output more than other attributes only because it has higher ranges, so it is important to apply feature scaling.

We applied min-max scaling on two columns
['rating_count_tot', 'rating_count_ver']

```
#feature scaling
scaler = MinMaxScaler()
data['rating_count_tot'] = scaler.fit_transform(np.array(data['rating_count_tot']).reshape(-1,1))
data['rating_count_ver'] = scaler.fit_transform(np.array(data['rating_count_ver']).reshape(-1,1))
#print("***** Scaling *****")
```

Features Used

Feature	Description
rating_count_tot	User Rating counts (for all version)
rating_count_ver	User Rating counts (for current version)
prime_genre	Primary Genre
sup_devices.num	Number of supporting devices
lang.num	Number of supported languages
ipadSc_urls.num	Number of screenshots shown for display

Classification Techniques

As we know that classification and regression are two major prediction problems which are usually dealt with Data Science and Machine Learning.

Classification is the process of finding or discovering a model or function which helps in separating the data into multiple categorical classes discrete values.

Regression is the process of finding a model or function for distinguishing the data into continuous real values instead of using classes or discrete values. It can also identify the distribution movement depending on the historical data

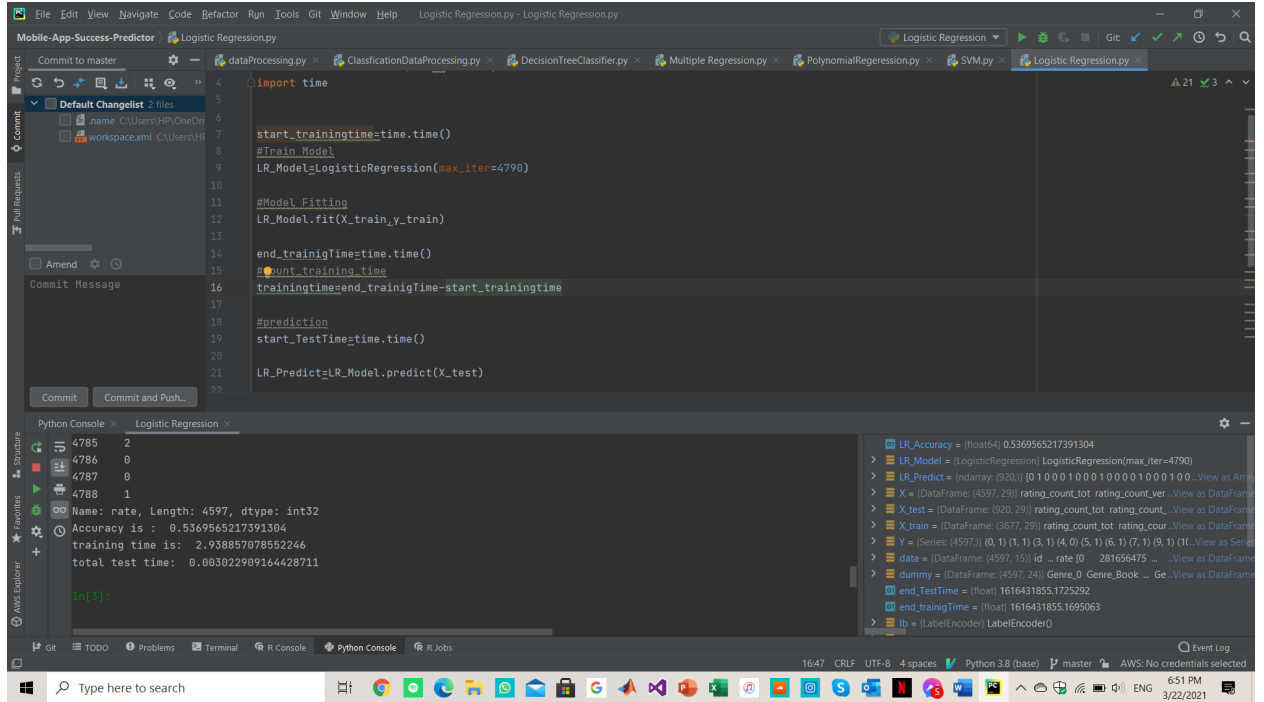
In classification , data is categorized under different labels according to some parameters given in input and then the labels are predicted for the data.

• Logistic Regression

It's a classification algorithm used to assign observations on a set of discrete classes , some examples of classification like email spam or not spam.

- Simple logistic regression =logistic regression with one predictor variable.
- Multiple logistic regression =logistic regression with multiple predictor variables.
- And it also transforms its output using the logistic sigmoid function to return a probability value.

Logistic Regression Results:



The screenshot displays a VS Code editor window with a project named "Mobile-App-Success-Predictor". The active file is "Logistic Regression.py". The code in the editor is as follows:

```
1 import time
2
3
4 start_trainingtime=time.time()
5 #Train_Model
6 LR_Model=LogisticRegression(max_iter=4790)
7
8 #Model_Fitting
9 LR_Model.fit(X_train,y_train)
10
11
12 end_trainingtime=time.time()
13 #count_training_time
14 trainingtime=end_trainingtime-start_trainingtime
15
16
17 #prediction
18 start_TestTime=time.time()
19
20 LR_Predict=LR_Model.predict(X_test)
21
```

The Python Console at the bottom shows the execution results:

```
Name: rate, Length: 4597, dtype: int32
Accuracy is : 0.5369565217391304
training time is: 2.938857078552246
total test time: 0.003022909164428711
```

On the right side of the console, the following variables are listed:

- LR_Accuracy = (float64) 0.5369565217391304
- LR_Model = (LogisticRegression) LogisticRegression(max_iter=4790)
- LR_Predict = (ndarray (920,)) [0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 ... View as Array]
- X = (DataFrame: (4597, 29)) rating_count_tot rating_count_ver ... View as DataFrame
- X_test = (DataFrame: (920, 29)) rating_count_tot rating_count_ver ... View as DataFrame
- X_train = (DataFrame: (3677, 29)) rating_count_tot rating_count_ver ... View as DataFrame
- Y = (Series: (4597,)) (0, 1) (1, 1) (3, 1) (4, 0) (5, 1) (6, 1) (7, 1) (9, 1) (11, ... View as Series
- data = (DataFrame: (4597, 15)) id ... rate [0 281656475 ... View as DataFrame
- dummy = (DataFrame: (4597, 24)) Genre_0 Genre_Book ... Ge ... View as DataFrame
- end_TestTime = (float) 1616431855.1725292
- end_trainingTime = (float) 1616431855.1695063
- lb = (LabelEncoder) LabelEncoder()

The status bar at the bottom indicates the file is "Logistic Regression.py" in "Python 3.8 (base)" environment, with "master" branch selected. The system clock shows 6:51 PM on 3/22/2021.

Bar Graph For Logistic regression:



Support Vector Machine

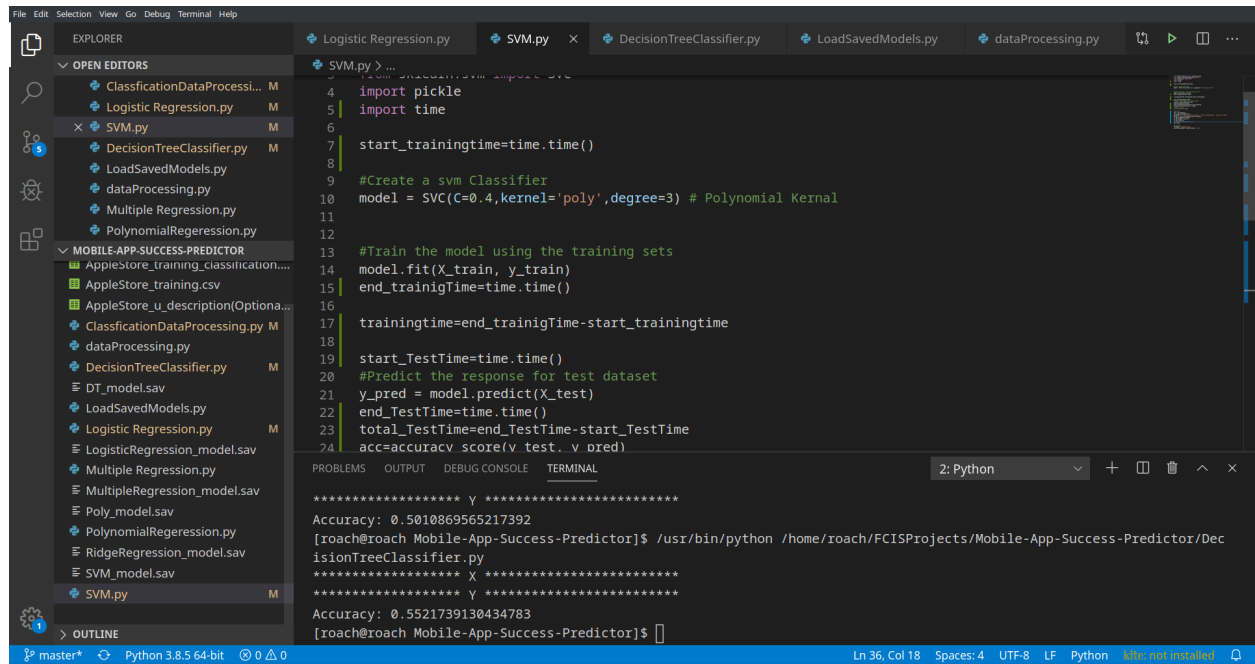
As we know SVM is supervised machine learning which can be used for both classification or regression problems. SVM is most used with linearly separable data.

It mostly used in classification problems, in this algorithm we plot each data item as a point in n-dimensional space (n-> number of features)

If the data isn't linear we perform a transformation using Kernel functions.

In this model we used SVM with $C=0.4$ and Polynomial Kernel Function of degree 3.

SVM Results:



The screenshot shows a VS Code editor with a project named 'MOBILE-APP-SUCCESS-PREDICTOR'. The Explorer sidebar on the left lists files including 'AppStore_training_classification...', 'AppleStore_training.csv', 'AppleStore_u_description(Optionala...', 'ClassificationDataProcessing.py', 'dataProcessing.py', 'DecisionTreeClassifier.py', 'DT_model.sav', 'LoadSavedModels.py', 'Logistic Regression.py', 'LogisticRegression_model.sav', 'Multiple Regression.py', 'MultipleRegression_model.sav', 'Poly_model.sav', 'PolynomialRegression.py', 'RidgeRegression_model.sav', 'SVM_model.sav', and 'SVM.py'. The 'SVM.py' file is open in the editor, showing the following code:

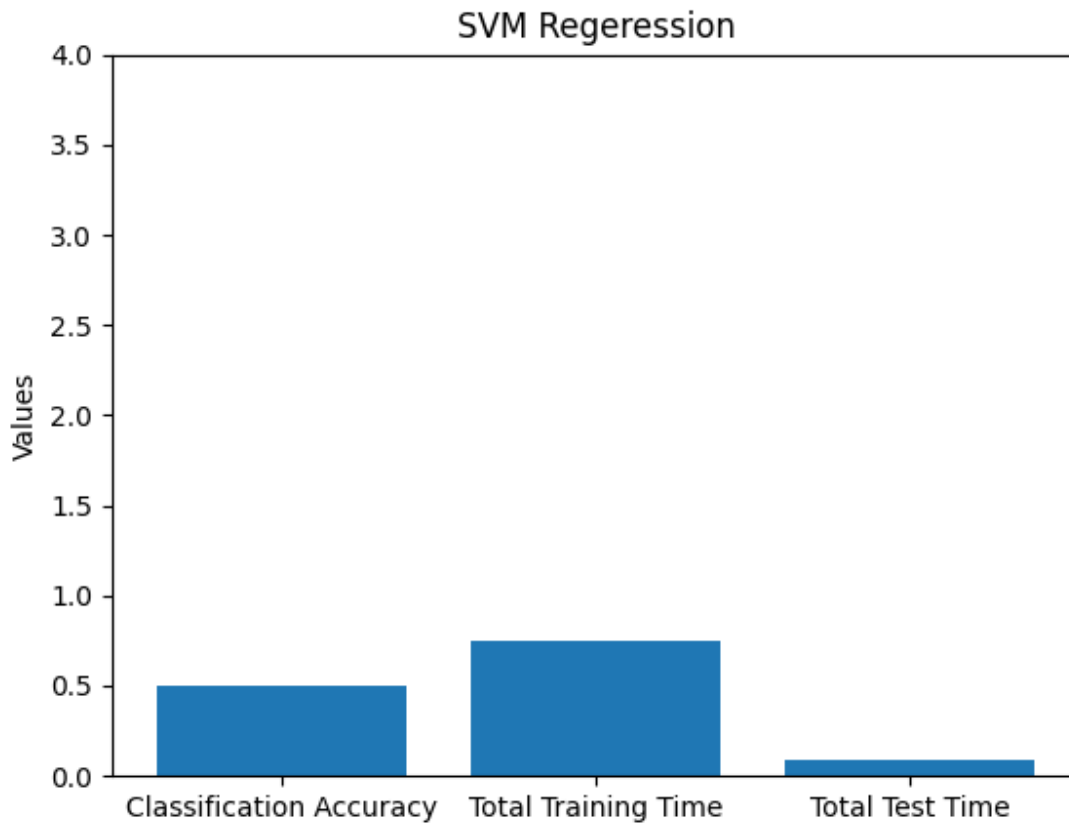
```
1 from sklearn.svm import SVC
2 import pickle
3 import time
4
5
6
7 start_trainingtime=time.time()
8
9 #Create a svm Classifier
10 model = SVC(C=0.4,kernel='poly',degree=3) # Polynomial Kernal
11
12
13 #Train the model using the training sets
14 model.fit(X_train, y_train)
15 end_trainigTime=time.time()
16
17 trainingtime=end_trainigTime-start_trainingtime
18
19 start_TestTime=time.time()
20 #Predict the response for test dataset
21 y_pred = model.predict(X_test)
22 end_TestTime=time.time()
23 total_TestTime=end_TestTime-start_TestTime
24 acc=accuracy_score(y_test, y_pred)
```

The terminal at the bottom shows the execution of the script:

```
Accuracy: 0.5010869565217392
[roach@roach Mobile-App-Success-Predictor]$ /usr/bin/python /home/roach/FCISProjects/Mobile-App-Success-Predictor/DecisionTreeClassifier.py
Accuracy: 0.5521739130434783
[roach@roach Mobile-App-Success-Predictor]$
```

The status bar at the bottom indicates the file is on line 36, column 18, using Python 3.8.5 64-bit.

SVM Plotted Bar Graph:



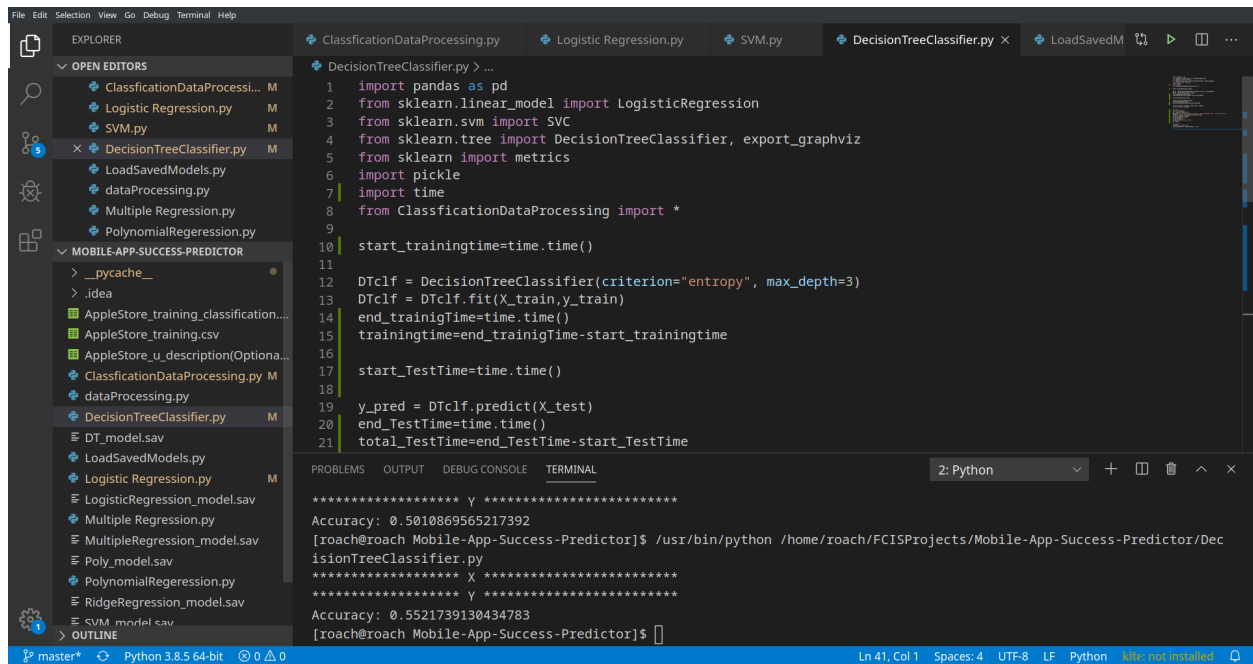
Decision Tree

Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning.

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making.

In this model we used a decision tree with max depth of 3 which resulted in approximate accuracy to other models with much less training and test time resulting in much better performance.

Decision Tree Results:



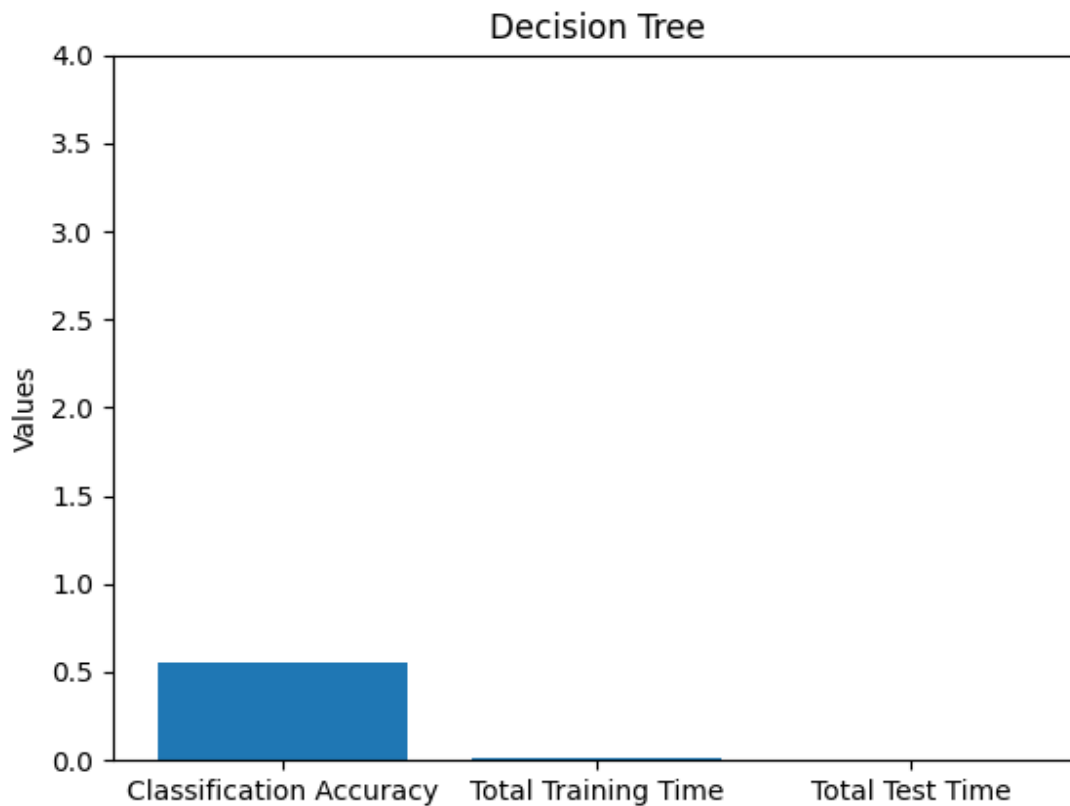
The screenshot displays a Jupyter Notebook environment with the following components:

- EXPLORER:** Shows the file structure of the project, including files like `ClassificationDataProcessing.py`, `Logistic Regression.py`, `SVM.py`, and `DecisionTreeClassifier.py`.
- Code Editor:** Contains the Python code for the `DecisionTreeClassifier.py` file. The code imports necessary libraries, loads data, and trains a `DecisionTreeClassifier` with a maximum depth of 3. It also calculates the training and testing times and accuracy.
- TERMINAL:** Shows the output of the code execution, including the accuracy of the model and the execution time.

```
1 import pandas as pd
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.svm import SVC
4 from sklearn.tree import DecisionTreeClassifier, export_graphviz
5 from sklearn import metrics
6 import pickle
7 import time
8 from ClassificationDataProcessing import *
9
10 start_trainingtime=time.time()
11
12 DTclf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
13 DTclf = DTclf.fit(X_train,y_train)
14 end_trainigTime=time.time()
15 trainingtime=end_trainigTime-start_trainingtime
16
17 start_TestTime=time.time()
18
19 y_pred = DTclf.predict(X_test)
20 end_TestTime=time.time()
21 total_TestTime=end_TestTime-start_TestTime
```

***** y *****
Accuracy: 0.5010869565217392
[roach@roach Mobile-App-Success-Predictor]\$ /usr/bin/python /home/roach/FCISProjects/Mobile-App-Success-Predictor/DecisionTreeClassifier.py
***** x *****
***** y *****
Accuracy: 0.5521739130434783
[roach@roach Mobile-App-Success-Predictor]\$

Decision Tree Plotted Bar Graph:



Conclusion:

- SVM Try to maximize the margin between the closest support vectors.
- Logistic Regression maximizes likelihood.
- Data preprocessing and good feature selection leads to better results with better accuracy.
- Accuracy of the Logistic regression is slightly better than SVM in this Model but training time is much larger while logistic regression has less test time.

- Decision Tree accuracy is close slightly less but training and test time are way less than SVM and LR in this model.