RATINGS PREDICTION PROJECT

Submitted by:

Komal Vijay Ghatvilkar

ACKNOWLEDGMENT

The success & outcome of this project were possible by the guidance and support from FlipRobo.

It was not possible to done without research from different machine learning sites and models on Google.

I referred DataTrained material for more information and some old machine learning projects that helped me completion of the project.

Thank you.....!!!

The internet has brought all things to our fingertips, from buying groceries to researching our next new automobile purchase. Once a place for posting a pretty website to promote your business, the internet is now evolving to be a forum where consumers evaluate products and services based on impressions and feedback from other, like-minded consumers.

It is easy to assume the importance of customer reviews, but nothing highlights more than objective data just how reviews are used and how they impact business. Statistics analyse how customers behave before and after using services or buying products, which can help develop plans to improve business.

Reviews not only have the power to influence consumer decisions but can strengthen a company's credibility. Reviews have the power to gain customer trust, and they encourage people to interact with the company. Customer interaction ultimately leads to improved profits for businesses.

I have collected the data from "Flipkart" for different electronics gadgets. The problem statement needs to predict ratings for the reviews which were written in the past and they don't have a rating. So, we have to build a machine learning model which can predict the rating by seeing the review. We have done the following analysis of the dataset where we Imported necessary libraries so that we can work on datasets with the Jupyter notebook.

```
- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns
- import warnings
- warnings.filterwarnings('ignore')
```

Data contains 61964 entries each having 4 variables. After reading the dataset I proceed with the EDA.

```
- df = pd.read_excel(r'C:\Users\HP\Desktop\RatingsPredictionData.xlsx')
- df.head()
```

Then, I removed the unwanted column and perform label encoder technique to convert the string into integer for further process.

```
- df.drop({'Unnamed: 0'},axis=1,inplace=True)

- from sklearn.preprocessing import LabelEncoder
- LE=LabelEncoder()
- df['Review_Summary']=LE.fit_transform(df['Review_Summary'])
- df['Product_Name']=LE.fit_transform(df['Product_Name'])
```

I checked the description of data with .info() method.

- `df.info()`

After .describe() done found the statistical description of data & found no null values so performed the task further.

- `df.describe()`
- `df.isnull().sum()`

With the correlation among all the columns checked the correlation and some data is positively correlated and some negatively correlated with each other.

- `df.corr()`

In data visualization done the following visualizations :

First used Correlation Matrix for showing the correlation between all columns with Heatmap.

From the output of correlation matrix, we can see that it is symmetrical i.e. the bottom left is same as the top right.

```
# Correlation Matrix
corr_mat=df.corr()
# Size of the canvas
plt.figure(figsize=[10,10])
#Plot Correlation Matrix
sns.heatmap(corr_mat,annot=True) # annot represnts each value encoded in heatmap
plt.title('Correlation Matrix')
plt.show()
```

The result of the Correlation Matrix is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/Correlation%20Matrix.png

Second, used Histogram for all dataset for visualizing the data individually.

The visualization plot shows that each variable distributed differently and as we can see data has categorical values, so used histogram for better understanding to show the distribution.

- `# plot histogram data vizualization`
- `df.hist(bins=20,figsize=(10,10))`
- `#plot showing`
- `plt.show()`

The result of Histogram is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/Histogram.png

Then I checked the skewness of the data. After that checked the outliers if any, found no outliers so didn't removed anything. I used boxplot to check the outliers in dataset.

The result of the Boxplot is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/Outliers.png

Then to show the toxic words and comments did the following task by using NLTK libraries and adding column for checking the final length of the reviews :-

```
- from nltk.stem import WordNetLemmatizer
- import nltk
- from nltk.corpus import stopwords
- import string

- df['length'] = df['Full_Review'].str.len()
- df.head()

- # Convert all messages to lower case
- df['Full_Review'] = df['Full_Review'].str.lower()

- # Replace email addresses with 'email'
- df['Full_Review'] = df['Full_Review'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
-                                  'emailaddress')

- # Replace URLs with 'webaddress'
- df['Full_Review']    =    df['Full_Review'].str.replace(r'^http\://[a-zA-Z0-9\-
  \.]+\.[a-zA-Z]{2,3}(/\S*)?$',
-                                   'webaddress')

- # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
- df['Full_Review'] = df['Full_Review'].str.replace(r'£|\$', 'dollers')
-
- # Replace 10 digit phone numbers (formats include paranthesis, spaces, no
  spaces, dashes) with 'phonenumber'
- df['Full_Review']     =     df['Full_Review'].str.replace(r'^\(?[\d]{3}\)?[\s-
  ]?[\d]{3}[\s-]?[\d]{4}$',
-                                   'phonenumber')
-
- # Replace numbers with 'numbr'
- df['Full_Review'] = df['Full_Review'].str.replace(r'\d+(\.\d+)?', 'numbr')


- df['Full_Review'] = df['Full_Review'].apply(lambda x: ' '.join(
-     term for term in x.split() if term not in string.punctuation))

- stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im',
  'dont', 'doin', 'ure'])
- df['Full_Review'] = df['Full_Review'].apply(lambda x: ' '.join(
-     term for term in x.split() if term not in stop_words))

- lem=WordNetLemmatizer()
- df['Full_Review'] = df['Full_Review'].apply(lambda x: ' '.join(
```

```
-    lem.lemmatize(t) for t in x.split()))

-   df['clean_length'] = df.Full_Review.str.len()
-   df.head()

-   df.length.sum()

-   df.clean_length.sum()
```

Then plotted graph to see the offensive words or bad words exists if any :-

```
#Getting sense of loud words which are offensive
from wordcloud import WordCloud
hams = df['Full_Review']
spam_cloud                                                              =
WordCloud(width=600,height=400,background_color='black',max_words=50).generat
e(' '.join(hams))
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

The result of the Wordcloud is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/WordCloud.png

Imported some libraries for model building :-

```
-   from sklearn.naive_bayes import MultinomialNB
-   from sklearn.model_selection import train_test_split
-   from     sklearn.metrics     import     accuracy_score,     confusion_matrix,
    classification_report,roc_curve,roc_auc_score,auc
-   from sklearn.model_selection import train_test_split
-   from                          sklearn.metrics                              import
    accuracy_score,classification_report,confusion_matrix,f1_score
-   from sklearn.linear_model import LogisticRegression
-   from sklearn.model_selection import cross_val_score,GridSearchCV
-   from sklearn.naive_bayes import MultinomialNB
-   from sklearn.tree import DecisionTreeClassifier
-   from sklearn.neighbors import KNeighborsClassifier
-   from                          sklearn.ensemble                             import
    RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
-   from sklearn.naive_bayes import GaussianNB
-   from sklearn.linear_model import LogisticRegression
-   from sklearn.svm import SVC
-   from sklearn.tree import DecisionTreeClassifier
-
```

Then Converted text into vectors using TF-IDF for model building task:-

```
-   from sklearn.feature_extraction.text import TfidfVectorizer
-   tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
-   features = tf_vec.fit_transform(df['Full_Review'])
-   x = features

-    df.shape
```

And then did the Train Test Split :-

```
- y=df['Rating']
- x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=150,test_size=
  .30)
- y_train.shape,y_test.shape
```

With the best suitable model building for categorical dataset performed The task and used various model bulding techniques like Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Classifier, AdaBoost Classifier & Kneighbors Classifier.

```
- # LogisticRegression
- LG = LogisticRegression(C=1, max_iter = 3000)
- LG.fit(x_train, y_train)
- y_pred_train = LG.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = LG.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- # DecisionTreeClassifier
- DT = DecisionTreeClassifier()
- DT.fit(x_train, y_train)
- y_pred_train = DT.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = DT.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #RandomForestClassifier
- RF = RandomForestClassifier()
- RF.fit(x_train, y_train)
- y_pred_train = RF.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = RF.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- # SVC - Support Vector Classifier
- from sklearn.svm import SVC
- svc=SVC()
- svc.fit(x_train,y_train)
- y_pred_train=svc.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = knn.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #AdaBoostClassifier
- ada=AdaBoostClassifier(n_estimators=100)
```

```
- ada.fit(x_train, y_train)
- y_pred_train = ada.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = ada.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #KNeighborsClassifier
- knn=KNeighborsClassifier(n_neighbors=9)
- knn.fit(x_train, y_train)
- y_pred_train = knn.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = knn.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))
```

Next did Cross Validation with "cross_val_score" for the models used & it shows the output :-

```
- from sklearn.model_selection import cross_val_score
- scr=cross_val_score(LG,x,y,cv=5)
- print("Cross Validation Score Of Logistic Regression Model :",scr.mean())
- scr1=cross_val_score(DT,x,y,cv=5)
- print("Cross Validation Score Of Decision Tree Model :",scr1.mean())
- scr2=cross_val_score(RF,x,y,cv=5)
- print("Cross Validation Score Of Random Forest Model :",scr2.mean())
- scr3=cross_val_score(ada,x,y,cv=5)
- print("Cross Validation Score Of Adaboost Model :",scr3.mean())
- scr4=cross_val_score(knn,x,y,cv=5)
- print("Cross Validation Score Of KNeighbors Model :",scr4.mean())
- scr5=cross_val_score(svc,x,y,cv=5)
- print("Cross Validation Score Of Support Vector Model :",scr5.mean())
```

Then checked the regularization with GridSearchCV to perform regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

```
- from sklearn.model_selection import GridSearchCV
- # Create parameters list to pass in GridSearchCV
- parameters={'max_features':['auto','sqrt','log2'],'max_depth':[4,5,6,7,8],'cri
  terion':['gini','entropy']}
- GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=5,scoring="accuracy")
- GCV.fit(x_train,y_train) # fitting the data in model
- GCV.best_params_ # Printing the best parameter found by GridSearchCV
- GCV_pred=GCV.best_estimator_.predict(x_test) # predicting with best parameters
- accuracy_score(y_test,GCV_pred) # checking final accuracy
```

With ROC Curve plot we found the multinominal distribution of data :-

```
- from sklearn.multiclass import OneVsRestClassifier
```

```
- from sklearn.datasets import make_classification
- from sklearn.metrics import roc_curve
- from sklearn.metrics import roc_auc_score
- # generate 2 class dataset
- x,y = make_classification(n_samples=200, n_classes=3, n_features=5,
  n_informative=3, random_state=0)
- # split into train/test sets
- x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.30,
  random_state=150)
- # fit model
- clf = OneVsRestClassifier(GCV.best_estimator_)
- clf.fit(x_train, y_train)
- pred = clf.predict(x_test)
- pred_prob = clf.predict_proba(x_test)
- # roc curve for classes
- fpr = {}
- tpr = {}
- thresh ={}
- n_class = 3
- for i in range(n_class):
-     fpr[i], tpr[i], thresh[i] = roc_curve(y_test,x_test[:,i], pos_label=i)
- # plotting
- plt.plot(fpr[0], tpr[0], linestyle='--',color='yellow', label='Rating 3 vs
  Rest')
- plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Rating 4 vs
  Rest')
- plt.plot(fpr[2], tpr[2], linestyle='--',color='blue', label='Rating 5 vs
  Rest')
- plt.title('ROC curve')
- plt.xlabel('False Positive Rate')
- plt.ylabel('True Positive rate')
- plt.legend(loc='lower right',fontsize='small')
- plt.savefig('Multiclass ROC',dpi=500)
- plt.show()

- # Conclusion:
- Attrition=np.array(y_test)
- Predicted=np.array(LG.predict(x_test))
- df_1=pd.DataFrame({'original':Attrition,'predicted':Predicted},index=range(le
  n(Attrition)))
- df_1

- # Model Saving
- import pickle
- filename = 'RatingsPrediction.pkl'
- pickle.dump(RF,open(filename,'wb'))
```

Finally came to the Conclusion as per the results found those are the best model is SVC and Random Forest Classifier showing the result for the dataset with approx. 80% accuracy. As per the understanding of the problem statement I used NLP method to know the reviews which having toxic or bad or offensive type of words which can straightly show that the rating for those reviews will not lie in good ratings but in our dataset

there are no such words found which is good. So, as per the data we can rely on 80% of the data which is correct.

Please find the GitHub links for ROC Curve to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/ROC%20Curve.png

Please find the GitHub link for Jupyter Notebook Solution of data collection to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/Ratings%20Prediction%20Project%20-%20Data%20Collection.ipynb

Please find the GitHub link for Jupyter Notebook Solution of data analysis to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Ratings%20Prediction%20Project/Ratings%20Prediction%20Project.ipynb

Thank You.....!!!