Rating Prediction

8/11/2021

Introduction

8/11/2021

- The problem which I have faced while doing this project is that to scrap the data from Amazon websites and to scrap the laptops reviews and Ratings as well with huge no of 38400 rows data .am face many time only one problem that is finding the Rating .

Data collection

- Firstly I have done the data extraction, for extracting I have used the amazon websites through that I have scraped the Rating and Reviews.
- And I got 38400 rows for Laptops

Analytical Problem Framing


```
#loading the Dataset
df_rating=pd.read_csv("D:/DATA_SCIENCE\I/intern Assignment/Review_rating/Review_Rating_final.csv")
```

```
df_rating.head()
```

	Unnamed: 0	Product_Review	Product_Rating
0	0	1. Price is up a bit . In this price we are ge...	1.0
1	1	If it is come with pre loaded ms office nd 165...	1.0
2	2	The delivery was well on time. The product was...	5.0
3	3	Amazing product with proper delivery!!Perfect ...	5.0
4	4	Lenovo Legion 5i is a decent rig for gaming as...	4.0

```
#cheking the Datatype
df_rating.dtypes
```

```
Unnamed: 0          int64
Product_Review     object
Product_Rating    float64
dtype: object
```

```
#Lets Drop the Unwanted data
df_rating.drop('Unnamed: 0',axis=1,inplace=True)
```

```
df_rating.head()
```

	Product_Review	Product_Rating
0	1. Price is up a bit . In this price we are ge...	1.0
1	If it is come with pre loaded ms office nd 165...	1.0
2	The delivery was well on time. The product was...	5.0
3	Amazing product with proper delivery!!Perfect ...	5.0
4	Lenovo Legion 5i is a decent rig for gaming as...	4.0

Data Pre-processing


```
#cheking the Datatype
df_rating.dtypes
```

```
Unnamed: 0          int64
Product_Review     object
Product_Rating    float64
dtype: object
```

```
#Lets Drop the Unwanted data
df_rating.drop('Unnamed: 0',axis=1,inplace=True)
```

```
df_rating.head()
```

	Product_Review	Product_Rating
0	1. Price is up a bit . In this price we are ge...	1.0
1	If it is come with pre loaded ms office nd 165...	1.0
2	The delivery was well on time. The product was...	5.0
3	Amazing product with proper delivery!!Perfect ...	5.0
4	Lenovo Legion 5i is a decent rig for gaming as...	4.0

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

```
#steaming
lemmatizer = WordNetLemmatizer()
def stemming(col):
    lema=lemmatizer.lemmatize(col,pos='a')
    return lema

```

```
#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(stemming(token))
    text=result
    return text
```

Data Pre-processing

- Finally we are calling all the Data Pre-processing function on a single column

```
def data_cleaning(df,col):
    Text_cleaning(df,col)
    Remove_Punctuatuation(df,col)
    Remove_Stop_Words(df,col)
    stemming(col)
    preprocess(col)
    return df
```

```
final=data_cleaning(df_rating,'Product_Review')
```

```
df_new=final
```

- Now We have your cleaned Data

Analysis

- We can see here which words are heavy when its comes to releted to rating “1”.

Analysis

- We can see here which words are heavy when its comes to releted to rating “2”

Analysis

- We can see here which words are heavy when its comes to releted to rating “3”.

Analysis

- We can see here which words are heavy when its comes to releted to rating “4”.

Analysis

- We can see here which words are heavy when its comes to releted to rating “5”.

Feature Extraction

- TFIDF vectorizer.

```
# 1. Convert text into vectors using TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer

tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(df_new['Product_Review'])

x = features
y = df_new[['Product_Rating']]
```

Model creation

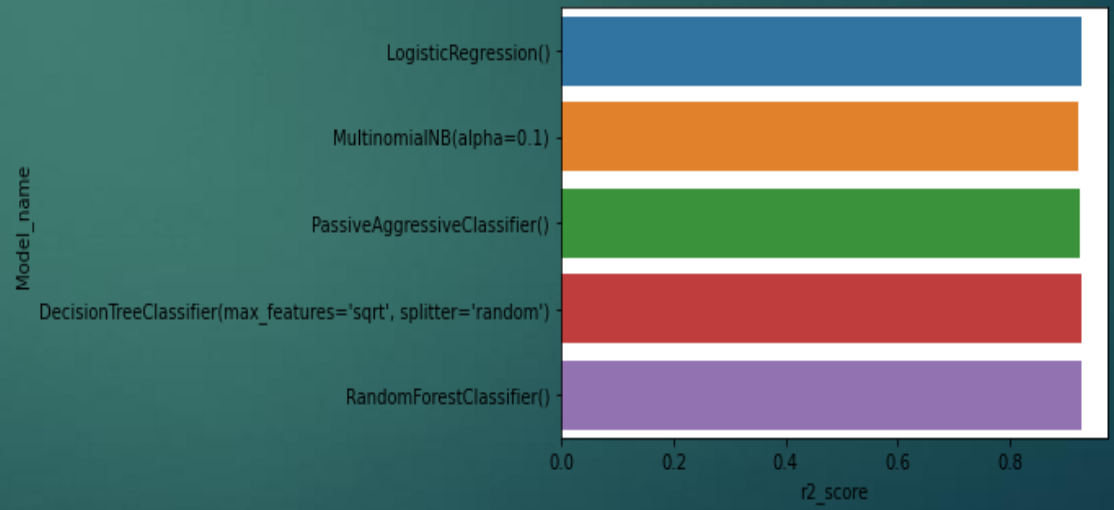
- Here We are making a function for model Training.

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
final_random_state=[]
final_r2score=[]
model=[]
def max_acc(rgr,x,y):
    max_acc=0
    model.append(rgr)
    for r in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r,test_size=0.20)
        rgr.fit(x_train,y_train)
        y_prd=rgr.predict(x_test)
        rc=accuracy_score(y_test,y_prd)
        if rc>max_acc:
            max_acc=rc
            final_r=r
    final_random_state.append(final_r)
    final_r2score.append(max_acc)
    print("max accuracy_ score coressponding to **→",final_r,"is**",max_acc*100)
```

Model Selection

- After trained our data in different model we got the result

	Model_name	r2_score	Random_State
0	LogisticRegression()	0.927604	73
1	MultinomialNB(alpha=0.1)	0.922396	43
2	PassiveAggressiveClassifier()	0.925260	81
3	DecisionTreeClassifier(max_features='sqrt', sp...	0.927604	73
4	(DecisionTreeClassifier(max_features='auto', r...	0.927604	73



Model finalize

- We got the result where Randomforest is working good so am finalize Randomforestclassifier as my final model.

```
# Based on the Above Score am going to finalize the model as RandomForest Classifier
RF=RandomForestClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=73,test_size=0.20)
RF.fit(x_train,y_train)
RF_pred=RF.predict(x_test)
print(accuracy_score(y_test,RF_pred))
print(confusion_matrix(y_test,RF_pred))
print(classification_report(y_test,RF_pred))
```

```
0.9276041666666667
[[2119    8    9   31   38]
 [  33  387    0    1   16]
 [  59    8  592   14   11]
 [  49    0    8 1451   60]
 [ 133    9   31   38 2575]]
              precision    recall  f1-score   support

         1.0       0.89      0.96      0.92      2205
         2.0       0.94      0.89      0.91       437
         3.0       0.93      0.87      0.89       684
         4.0       0.95      0.93      0.94      1568
         5.0       0.95      0.92      0.94      2786

    accuracy                           0.93      7680
   macro avg       0.93      0.91      0.92      7680
weighted avg       0.93      0.93      0.93      7680
```

Actual vs Predict


```
#here we can check the predicted values
test=pd.DataFrame(data=y_test,)
test['Predicted values']=RF_pred

test.to_csv('Ratings_Predict.csv')

test
```

	Product_Rating	Predicted values
6379	5.0	5.0
17071	5.0	5.0
3402	4.0	4.0
24338	4.0	4.0
7646	2.0	2.0
...	...	...
4609	5.0	5.0
28224	3.0	3.0
15873	5.0	5.0
35155	5.0	5.0
22012	5.0	2.0

CONCLUSION

- **Key Findings and Conclusions of the Study**
- **The key findings that I have find that I have scraped it from only one websites due to dead line I was able to scrap it .if I could scrap more websites we will get more better model prediction.**
- **By using 38400 data we for two best models Random Forest Classifier and Decision Tree Classifier. Because of limited data I haven't go for sampling only just used stratify method to balance the data.**
- **Limitations of this work and Scope for Future Work.**
- In some algorithms where was taking to much time to execute but it was executed it in better way. because of that laptops where getting hang and as we accept we got better score in every