Model Development Phase Template

Date	July 2024
Team ID	740107
Project Title	The Language Of Youtube: A Text Classification Approach To Video Descriptions
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

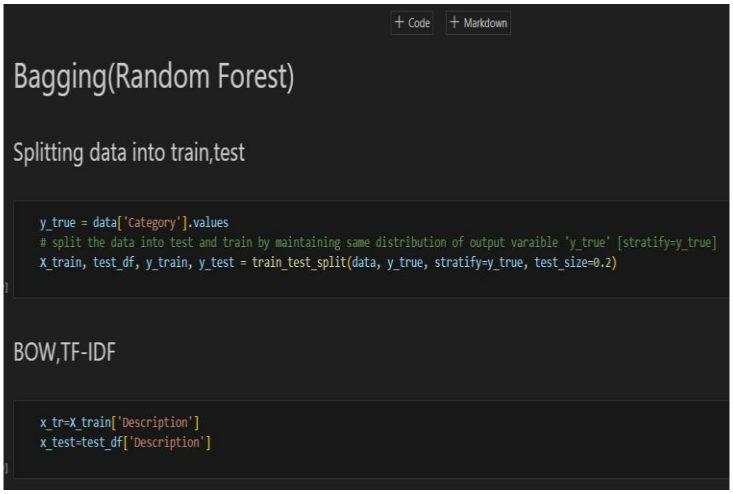
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

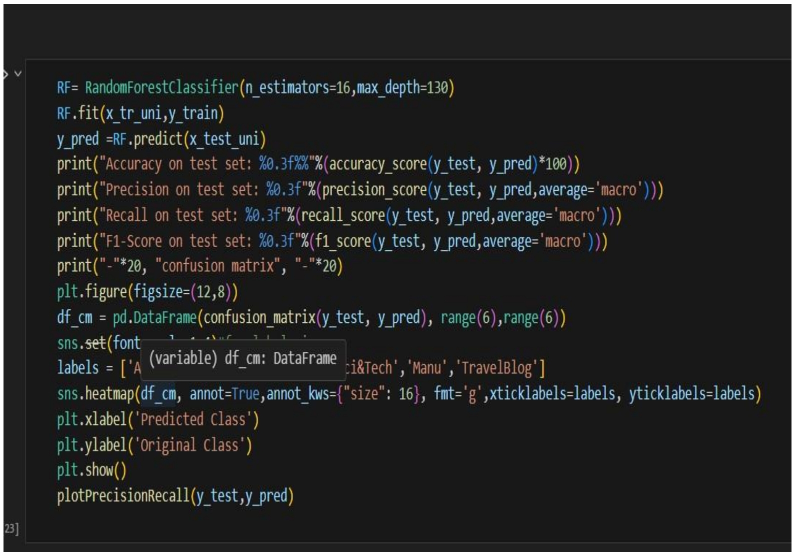
Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

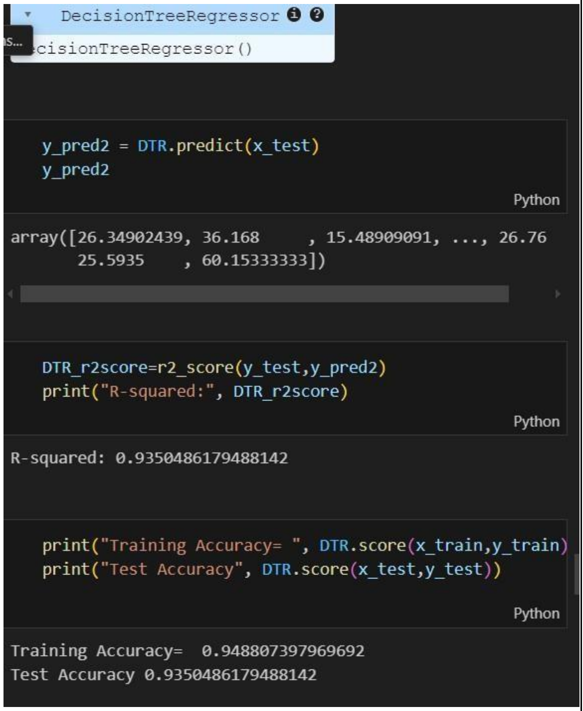
Model	**Summary**	**Training and Validation Performance Metrics**

Model Validation and Evaluation Report (5 marks):

Model 1	Logistic regression model typically include accuracy, precision, recall, r2_score to evaluate its predictive performance and generalization capability.	
		Bagging(Random Forest)
		Splitting data into train,test
		`y_true = data['Category'].values`
		`# split the data into test and train by maintaining same distribution of output varaible 'y_true' [stratify=y_true]`
		`X_train, test_df, y_train, y_test = train_test_split(data, y_true, stratify=y_true, test_size=0.2)`
		BOW,TF-IDF
		`x_tr=X_train['Description']`
		`x_test=test_df['Description']`

Model 2	Random forest classifier model often encompass accuracy, precision, recall, r2_score to measure its prediction quality and robustness.	

```
RF= RandomForestClassifier(n_estimators=16,max_depth=130)
RF.fit(x_tr_uni,y_train)
y_pred =RF.predict(x_test_uni)
print("Accuracy on test set: %0.3f%%"%(accuracy_score(y_test, y_pred)*100))
print("Precision on test set: %0.3f"%(precision_score(y_test, y_pred,average='macro')))
print("Recall on test set: %0.3f"%(recall_score(y_test, y_pred,average='macro')))
print("F1-Score on test set: %0.3f"%(f1_score(y_test, y_pred,average='macro')))
print("-"*20, "confusion matrix", "-"*20)
plt.figure(figsize=(12,8))
df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), range(6),range(6))
sns.~~set~~(font
labels = ['A                            ci&Tech','Manu','TravelBlog']
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g',xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
plotPrecisionRecall(y_test,y_pred)
```

Model 3	Decision tree classifier model commonly include accuracy, precision, recall, r2_score which help assess the model's prediction	

```
cisionTreeRegressor()
```

```
y_pred2 = DTR.predict(x_test)
y_pred2
```

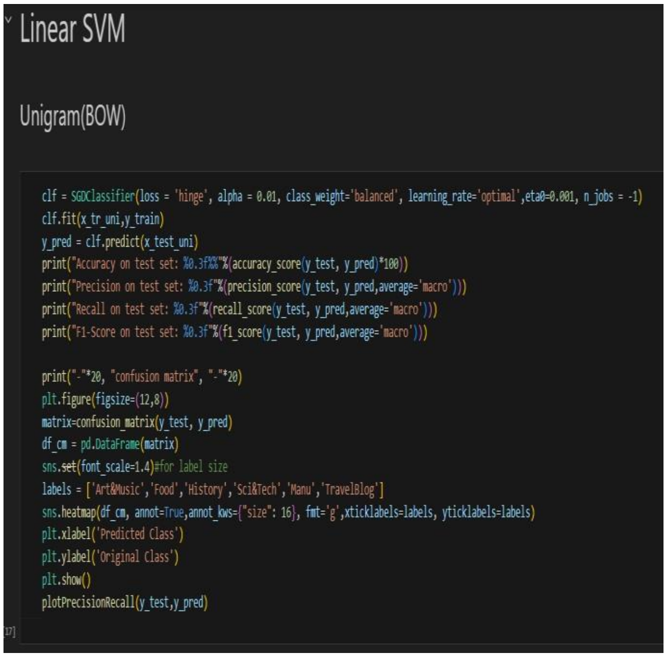
```
array([26.34902439, 36.168     , 15.48909091, ..., 26.76
       25.5935    , 60.15333333])
```

```
DTR_r2score=r2_score(y_test,y_pred2)
print("R-squared:", DTR_r2score)
```

```
R-squared: 0.9350486179488142
```

```
print("Training Accuracy= ", DTR.score(x_train,y_train)
print("Test Accuracy", DTR.score(x_test,y_test))
```

```
Training Accuracy=  0.948807397969692
Test Accuracy 0.9350486179488142
```

Model 4	accuracy and generalizability. Linear Support Vector Machines (SVM): A supervised learning model that finds the hyperplane that best divides a dataset into classes. • **Use Case**: Effective in high-dimensional spaces and commonly used for text classification.	

Linear SVM

Unigram(BOW)

```python
clf = SGDClassifier(loss = 'hinge', alpha = 0.01, class_weight='balanced', learning_rate='optimal',eta0=0.001, n_jobs = -1)
clf.fit(x_tr_uni,y_train)
y_pred = clf.predict(x_test_uni)
print("Accuracy on test set: %0.3f%%"%(accuracy_score(y_test, y_pred)*100))
print("Precision on test set: %0.3f"%(precision_score(y_test, y_pred,average='macro')))
print("Recall on test set: %0.3f"%(recall_score(y_test, y_pred,average='macro')))
print("F1-Score on test set: %0.3f"%(f1_score(y_test, y_pred,average='macro')))

print("-"*20, "confusion matrix", "-"*20)
plt.figure(figsize=(12,8))
matrix=confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(matrix)
sns.set(font_scale=1.4)#for label size
labels = ['Art&Music','Food','History','Sci&Tech','Manu','TravelBlog']
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g',xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
plotPrecisionRecall(y_test,y_pred)
```