

## Model Development Phase Template

Date	July 2024
Team ID	739890
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.

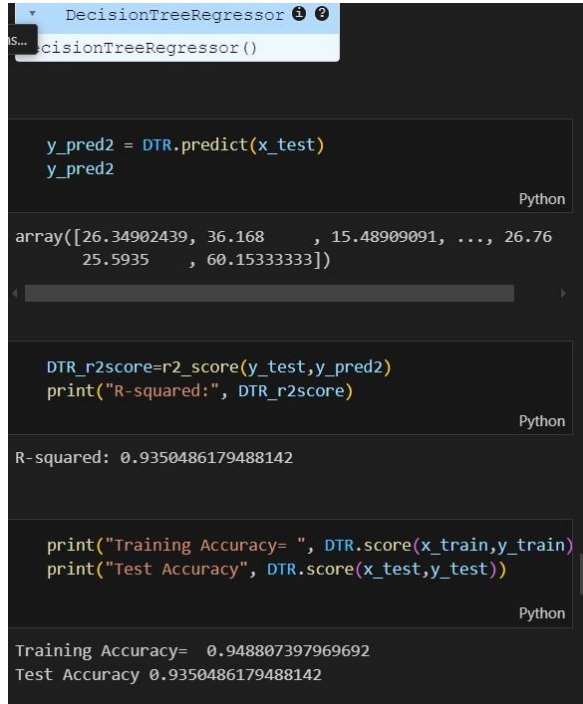
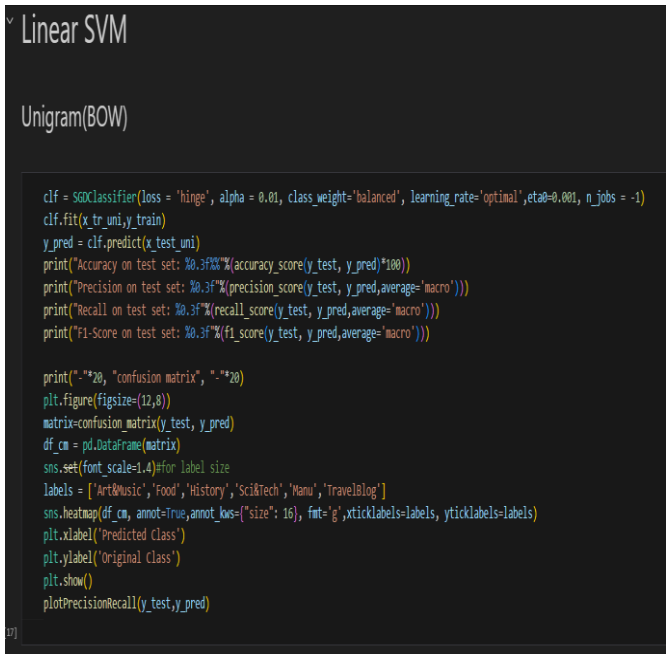
#### Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

#### Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

<p>Model 1</p>	<p>Logistic regression model typically include accuracy, precision, recall, r2_score to evaluate its predictive performance and generalization capability.</p>	<div data-bbox="646 226 1367 709"> <div> + Code + Markdown </div> <h3>Bagging(Random Forest)</h3> <p>Splitting data into train,test</p> <pre> y_true = data['Category'].values # split the data into test and train by maintaining same distribution of output variable '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) </pre> <h3>BOW,TF-IDF</h3> <pre> x_tr=X_train['Description'] x_test=test_df['Description'] </pre> </div>
<p>Model 2</p>	<p>Random forest classifier model often encompass accuracy, precision, recall, r2_score to measure its prediction quality and robustness.</p>	<div data-bbox="646 1087 1429 1633"> <pre> 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 (variable) df_cm: DataFrame', 'i&amp;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) </pre> </div>

Model 3	<p>Decision tree classifier model commonly include accuracy, precision, recall, r2_score which help assess the model's prediction accuracy and generalizability.</p>	 <pre> DecisionTreeRegressor DecisionTreeRegressor()  y_pred2 = DTR.predict(x_test) y_pred2 Python 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) Python R-squared: 0.9350486179488142  print("Training Accuracy= ", DTR.score(x_train,y_train)) print("Test Accuracy", DTR.score(x_test,y_test)) Python Training Accuracy=  0.948807397969692 Test Accuracy 0.9350486179488142 </pre>
Model 4	<p>Linear Support Vector Machines (SVM): A supervised learning model that finds the hyperplane that best divides a dataset into classes.</p> <ul style="list-style-type: none"> <li>• <b>Use Case:</b> Effective in high-dimensional spaces and commonly used for text classification.</li> </ul>	 <pre> Linear SVM  Unigram(BOW)  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: %.3f%%"%(accuracy_score(y_test, y_pred)*100)) print("Precision on test set: %.3f%%"(precision_score(y_test, y_pred,average='macro')))) print("Recall on test set: %.3f%%"(recall_score(y_test, y_pred,average='macro')))) print("F1-Score on test set: %.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&amp;Music','Food','History','Sci&amp;Tech','Manu','Travel&amp;log'] 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) </pre>

