# Model Optimization and Tuning 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 |

## Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

**Hyperparameter Tuning Documentation (8 Marks):**

| Model | Tuned Hyperparameters |
|---|---|
| Linear Regression | - |
| Random Forest Regressor | - |
| Decision Tree Regressor | - |

| SVM | - |
|-----|---|
|     |   |

| Logistic Regression | #importing the library for grid search<br>from sklearn.model_selection import GridSearchCV<br><br>The 'lr_param_grid' specifies different values for regularization strength (C), solvers (solver), and penalty types (penalty). GridSearchCV (lr_cv) is employed with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy"). The process uses all available CPU cores (n_jobs=-1) for parallel processing and provides verbose output (verbose=True) to track progress.<br><br>LOGISTIC REGRESSION HYPER PARAMETER TUNNING<br><br>`[54]` `#finding the grid search cv for logistic regression`<br>`lr=LogisticRegression(n_jobs=-1,random_state=0)`<br>`lr_param_grid={`<br>`    'C':[0.1,0.5,1,5,10],`<br>`    'solver':['liblinear','saga'],`<br>`    'penalty':['l1','l2']`<br>`}`<br>`lr_cv=GridSearchCV(lr,lr_param_grid,cv=5,scoring="accuracy",n_jobs=-1,verbose=T`<br>`lr_cv.fit(x_train,y_train)`<br><br>`Fitting 5 folds for each of 20 candidates, totalling 100 fits`<br>`/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1211:`<br>`  warnings.warn(`<br>`        GridSearchCV`<br>`▸ estimator: LogisticRegression`<br>`     ▸ LogisticRegression` |

| Random Forest | The parameter grid (rfc_param_grid) for hyperparameter tuning. It specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum number of features considered for splitting (max_features). GridSearchCV (rfc_cv) is employed with 3-fold cross-validation (cv=3), evaluating model performance based on accuracy (scoring="accuracy"). |
|---|---|

```python
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_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)
```

| Decision Tree | The parameters (params) define a grid for hyperparameter tuning of the Decision Tree Classifier (DecisionTreeClassifier), including max_depth, min_samples_leaf, and criterion ('gini' or 'entropy'). GridSearchCV (dec_cv) is used with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy") |
|---|---|

DECISION TREE CLASSIFIER-HYPER PARAMETER TUNNING

```python
[68] #finding grid search cv for decision tree classifier
     dec=DecisionTreeClassifier(random_state=42)
     params={ 'max_depth': [2, 3, 5, 10, 20],
             'min_samples_leaf': [5, 10, 20, 50, 100],
             'criterion': ['gini', 'entropy']
     }
     dec_cv=GridSearchCV(dec,param_grid=params,cv=5,n_jobs=-1,scoring="accuracy",verbose=3)
     dec_cv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
            GridSearchCV
  ▸ estimator: DecisionTreeClassifier
       ▸ DecisionTreeClassifier
```

| SDGClassifier | SGDClassifier is a popular linear classifier in machine learning that uses stochastic gradient descent (SGD) to optimize the loss function. It is well-suited for handling large-scale and sparse datasets, and can be used for both binary and multiclass classification problems. SGDClassifier allows for tuning of hyperparameters such as the learning rate, regularization, and loss function, making it a flexible and powerful tool for classification tasks. |
|---|---|

```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)
```

# Final Model Selection Justification (2 Marks):

| Final Model | Reasoning |
|---|---|

| | |
|---|---|
| Linear SVM model | The results show that all models have achieved good accuracy, with the Random Forest Classifier using BOW achieving the highest accuracy of 86.478%. However, the SGDClassifier using TF-IDF achieved the highest precision, recall, and F1-score of 0.917. This suggests that the SGDClassifier using TF-IDF is better at correctly identifying positive instances (precision), identifying all positive instances (recall), and balancing both metrics (F1-score) than the other models |

### Support Vector Machine

| Model | hyper parameter | F1-score cv | F1-score test | Precision test | Recall test | Accuracy Test |
|---|---|---|---|---|---|---|
| unigram | 0.01 | 0.927 | 0.907 | 0.908 | 0.907 | 90.836% |
| TF-IDF | 0.0001 | 0.922 | 0.920 | 0.920 | 0.920 | 92.138% |

### Random Forest

| Model | n_estimators | Max Depth | F1-score cv | F1-score test | Precision test | Recall test | Accuracy Test |
|---|---|---|---|---|---|---|---|
| unigram | 16 | 130 | 0.868 | 0.858 | 0.860 | 0.858 | 86.119% |
| TF-IDF | 20 | 190 | 0.872 | 0.849 | 0.854 | 0.848 | 85.085% |