Project Development Phase
Model Performance Test

Date	22 November 2023
Team ID	591830
Project Name	Endocrine Elegance: Classifying Thyroid Disorders with Precision
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot

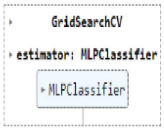
1.	Metrics	**Regression Model:**
		MAE - , MSE - , RMSE - , R2 score -
		
		Classification Model:
		Confusion Matrix - , Accuray Score- & Classification Report -
			Regression Model:
			In [90]:
			`import pandas as pd`
			`from sklearn.model_selection import train_test_split`
			`from sklearn.ensemble import RandomForestRegressor`
			`from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score`
			`import numpy as np`
			
			`# Load the dataset`
			`data = pd.read_csv('thyroidDF.csv')`
			
			`# Split the data into training and testing sets`
			`X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)`
			
			`# Initialize and train the RandomForestRegressor`
			`regressor = RandomForestRegressor(random_state=42)`
			`regressor.fit(X_train, y_train)`
			
			`# Make predictions on the test set`
			`y_pred = regressor.predict(X_test)`
			
			`# Calculate Mean Absolute Error (MAE)`
			`mae = mean_absolute_error(y_test, y_pred)`
			
			`# Calculate Mean Squared Error (MSE)`
			`mse = mean_squared_error(y_test, y_pred)`
			
			`# Calculate Root Mean Squared Error (RMSE)`
			`rmse = np.sqrt(mse)`
			
			`# Calculate R2 Score`
			`r2 = r2_score(y_test, y_pred)`
			
			`print(f'Mean Absolute Error: {mae}')`
			`print(f'Mean Squared Error: {mse}')`
			`print(f'Root Mean Squared Error: {rmse}')`
			`print(f'R2 Score: {r2}')`
			
			Mean Absolute Error: 1.474948228828337
			Mean Squared Error: 16.752909809264306
			Root Mean Squared Error: 4.093031860279652
			R2 Score: 0.7760056042002713
			
			Classification Model:

In [92]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('thyroidDF.csv')


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Initialize the RandomForestClassifier
classifier = RandomForestClassifier(random_state=42)

# Train the model
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)

# Accuracy Score
acc_score = accuracy_score(y_test, y_pred)
print(f'Accuracy Score: {acc_score}')

# Classification Report
class_report = classification_report(y_test, y_pred)
print('Classification Report:')
print(class_report)
```

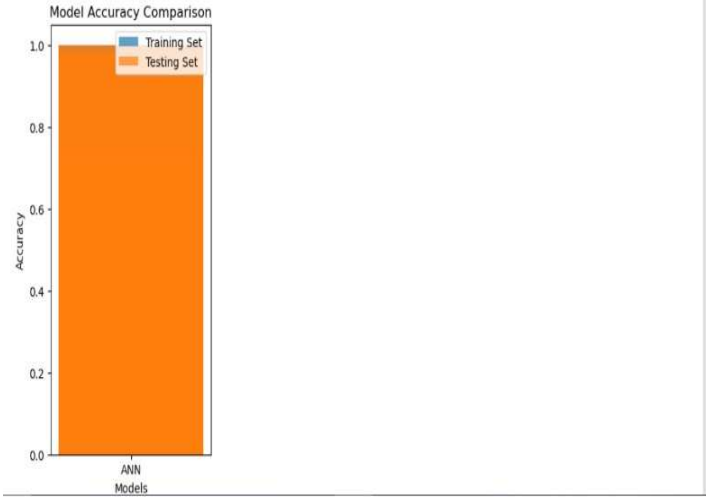
File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Markdown

```
Confusion Matrix:
[[1289    3    1    0    1    0    1    0    0   10    0   10    0    5
     0    0    1    0    1    5    1]
 [   2   15    0    0    0    0    0    0    0    1    0    0    0    0
     0    0    0    0    0    3    0]
 [   0    1    7    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    2    0]
 [   4    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0   38    0    1    0    1    0    0    0    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    1    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0   69    0    0    0    0    0    0    0
     0    0    0    0    0    0    0]
 [   1    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    6    0    0    0    0
     0    0    0    0    0    0    0]
 [  21    3    0    0    0    0    0    0    0   57    0    0    0    0
     0    0    1    0    0    0    0]
 [   6    0    0    0    0    0    0    0    0    0    0    6    0    0
     0    0    0    0    0    0    0]
 [   8    0    0    0    0    0    0    0    0    0    0    0   98    0
     0    0    0    0    0    0    0]
 [   1    0    0    0    0    0    0    0    0    0    0    0    0    1
     0    0    0    0    0    0    0]
 [   4    1    0    0    0    0    0    0    0    0    1    0    0    0
    15    0    6    0    0    1    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0   25    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    6    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     3    0    0   17    0    0    0]
 [   0    0    0    0    2    0    0    0    0    0    0    0    0    0
     0    0    0    2    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    1    0
     0    0    0    0    2    0    0]
 [  15    1    1    0    0    0    0    0    0    0    0    0    1    0
     0    0    0    0    0   27    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   22]]
Accuracy Score: 0.9275204359673025
Classification Report:
              precision    recall  f1-score   support

         0.0       0.95      0.97      0.96      1328
         1.0       0.62      0.71      0.67        21
         2.0       0.78      0.70      0.74        10
         3.0       0.00      0.00      0.00         4
         9.0       0.90      0.95      0.93        40
        10.0       0.00      0.00      0.00         1
        11.0       0.97      1.00      0.99        69
        12.0       0.00      0.00      0.00         1
        13.0       0.86      1.00      0.92         6
        16.0       0.83      0.70      0.75        82
        17.0       1.00      0.50      0.67        12
        18.0       0.89      0.92      0.91       106
        19.0       1.00      0.50      0.67         2
        20.0       0.65      0.54      0.59        28
        22.0       1.00      1.00      1.00        25
        24.0       1.00      1.00      1.00         6
        25.0       0.68      0.85      0.76        20
        26.0       1.00      0.50      0.67         4
        29.0       0.67      0.67      0.67         3
        30.0       0.71      0.60      0.65        45
        31.0       0.96      1.00      0.98        22

    accuracy                           0.93      1835
   macro avg       0.74      0.67      0.69      1835
weighted avg       0.92      0.93      0.92      1835
```

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Hyperparameter Tuning
			Compare the model
			`import pandas as pd`
			`from sklearn.model_selection import train_test_split, GridSearchCV`
			`from sklearn.ensemble import RandomForestClassifier`
			`from sklearn.svm import SVC`
			`from xgboost import XGBClassifier`
			`from sklearn.neural_network import MLPClassifier`
			`import matplotlib.pyplot as plt`
			`# Hyperparameter grids`
			`param_grids = {`
			`'Random Forest': {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]},`
			`'SVC': {'C': [1, 10, 100], 'kernel': ['linear', 'rbf']},`
			`'XGBoost': {'n_estimators': [50, 100, 200], 'max_depth': [3, 6, 9]},`
			`'ANN': {'hidden_layer_sizes': [(50,), (100,), (50, 50)], 'alpha': [0.0001, 0.001, 0.01]}`
			`}`
			`models = {'Random Forest': rf_model, 'SVC': svc_model, 'XGBoost': xgb_model, 'ANN': ann_model}`
			`y_train = y_train.astype(int)`
			`# Hyperparameter tuning and training`
			`for model_name, model in models.items():`
			`param_grid = param_grids[model_name]`
			`grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', n_jobs=-1)`
			`grid_search.fit(X_train, y_train)`
			`C:\Users\Leena\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:700: UserWarning: The least populated class as only 1 members, which is less than n_splits=5.`
			`warnings.warn(`
			 
			`best_model = grid_search.best_estimator_`
			`# Evaluate on training set`
			`train_acc = best_model.score(X_train, y_train)`
			`accuracy_scores_train.append(train_acc)`
			`# Evaluate on test set`
			`test_acc = best_model.score(X_test, y_test)`
			`accuracy_scores_test.append(test_acc)`
			`# Print results for each model`
			`print(f'{model_name} - Best Parameters: {grid_search.best_params_}')`
			`print(f'{model_name} - Training Accuracy: {train_acc:.4f}')`
			`print(f'{model_name} - Test Accuracy: {test_acc:.4f}')`
			`print()`
			`ANN - Best Parameters: {'alpha': 0.001, 'hidden_layer_sizes': (50, 50)}`
			`ANN - Training Accuracy: 0.9849`
			`ANN - Test Accuracy: 0.9738`

```
plt.figure(figsize=(3, 6))
plt.bar(model_name, accuracy_scores_train, label='Training Set', alpha=0.7)
plt.bar(model_name, accuracy_scores_test, label='Testing Set', alpha=0.7)

plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.legend()
plt.show()
```



Validation Method

```
In [91]: import pandas as pd
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import make_scorer, mean_squared_error
import numpy as np

# Load the dataset
data = pd.read_csv('thyroidDF.csv')


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Initialize the RandomForestRegressor
regressor = RandomForestRegressor(random_state=42)

# Define a custom scoring function (you can choose the metric you're interested in)
scorer = make_scorer(mean_squared_error, greater_is_better=False)

# Perform cross-validation (e.g., using 5-fold cross-validation)
cv_scores = cross_val_score(regressor, X_train, y_train, cv=5, scoring=scorer)

# Calculate mean and standard deviation of the cross-validation scores
mean_cv_score = np.mean(cv_scores)
std_cv_score = np.std(cv_scores)

print(f'Mean Cross-Validation Score: {mean_cv_score}')
print(f'Standard Deviation of Cross-Validation Scores: {std_cv_score}')

# Train the model on the full training set
regressor.fit(X_train, y_train)

# Evaluate the model on the test set
y_pred = regressor.predict(X_test)
test_score = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error on Test Set: {test_score}')

Mean Cross-Validation Score: -13.465419049376939
Standard Deviation of Cross-Validation Scores: 0.8274871413731842
Mean Squared Error on Test Set: 16.752909809264306
```