Name: **Muhammad Nabeel**

Assignment Type: **Internship Task**

Supervisor/Instructor: **Dr. Fehmida Usmani**

CMS: **413522**

Task: **Train a classification-model on a univariate Time-series data with Accuracy of 95% or above.**

Data Description:

The dataset contains 20,468 entries of data containing two column features and one output labels.

The features columns are Time in seconds of the respective dataset and other is the SOPAS feature given. The output label is the either value of (0,1,2 and 3) thus It was a classification problem.

Data Preprocessing:

For data preprocessing, no missing data was found. Further, feature engineering was not possible on the given dataset.

Data Splitting:

For the particular model, I have chosen 10 percent for test-set and 90 percent for training set.

Model Training:

- **Choice of Model:** Many different models were initially tested for this problem and they are described below:
  CNN: Convolution Neural Network was trained on this data and It yielded maximum of 76 percent of efficiency by various Hyperparameter tuning.
  ANN: A simple ANN was also trained and it also yielded maximum accuracy of 78 percent.
  Decision Trees: Decision Trees yielded accuracy of 81 percent on the given dataset.

KNN : Knn also yielded accuracy of approximate 80 percent.
Random Forests: This model when adjusted with different hyperparameters gave maximum accuracy of 85 percent and it was adopted as the final model.

- Hyperparameters Used:
  Tree Depth was left un-initialized as to give depth until which leaf nodes doesn't become pure.
  N-Parameter value of 200 is set as it gave most optimal results.

Accuracy Obtained:

After trying out different Architectures with various reports, A maximum **Accuracy of 85 percent** was achieved via the use of Random Forests.

Performance Metrics:

A separate pdf for Performance Metrics has been attached in the zip which.

Initial System Generated Stats:

```
Confusion Matrix (Test):
[[7230  132   30  791]
 [  96 1551  311   89]
 [   6  112 2115  498]
 [ 699   12  348 6449]]

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      8183
           1       0.86      0.76      0.80      2047
           2       0.75      0.77      0.76      2731
           3       0.82      0.86      0.84      7508

    accuracy                           0.85     20469
   macro avg       0.83      0.82      0.83     20469
weighted avg       0.85      0.85      0.85     20469
```

Further Stats of the data are all shared in a separate PDF.

Code Snippet:

```
# -*- coding: utf-8 -*-
"""
```

```
@author: Muhammad Nabeel
"""

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score, f1_score
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.backends.backend_pdf import PdfPages
import joblib

data = pd.read_csv('C:/Users/Nemro Neno/Desktop/Dataset_timeseries.csv')


X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=80)

rf = RandomForestClassifier(n_estimators=200, random_state=80, verbose=2,n_jobs=-
1)

rf.fit(X_train, y_train)

# Save the trained model
model_path = 'C:/Users/Nemro Neno/Desktop/rf_classifier_model.pkl'
joblib.dump(rf, model_path)
print(f'Model saved to {model_path}')

# Make predictions
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)

# Calculate metrics
metrics = {
    'Train': {
        'Accuracy': accuracy_score(y_train, y_pred_train),
        'Precision': precision_score(y_train, y_pred_train, average='weighted'),
        'Recall': recall_score(y_train, y_pred_train, average='weighted'),
        'F1 Score': f1_score(y_train, y_pred_train, average='weighted')
    },
```

```
    'Test': {
        'Accuracy': accuracy_score(y_test, y_pred_test),
        'Precision': precision_score(y_test, y_pred_test, average='weighted'),
        'Recall': recall_score(y_test, y_pred_test, average='weighted'),
        'F1 Score': f1_score(y_test, y_pred_test, average='weighted')
    }
}
print("Confusion Matrix (Test):")
print(confusion_matrix(y_test, y_pred_test))
print("\nClassification Report (Test):")
print(classification_report(y_test, y_pred_test))

pdf_path = 'C:/Users/Nemro Neno/Desktop/Model_Evaluation_Report.pdf'
pdf = PdfPages(pdf_path)

plt.figure(figsize=(12, 8))
sns.heatmap(confusion_matrix(y_test, y_pred_test), annot=True, fmt='d',
cmap='Blues', cbar=False)
plt.title('Confusion Matrix - Test Set')
plt.xlabel('Predicted')
plt.ylabel('Actual')
pdf.savefig()
plt.close()

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Model Performance Metrics Comparison', fontsize=16)

metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
for I, metric in enumerate(metric_names):
    ax = axes[i//2, i%2]
    ax.bar(['Train', 'Test'], [metrics['Train'][metric],
metrics['Test'][metric]], color=['blue', 'orange'])
    ax.set_ylim(0, 1)
    ax.set_title(metric)
    ax.set_ylabel('Score')
    for j, value in enumerate([metrics['Train'][metric],
metrics['Test'][metric]]):
        ax.text(j, value, f'{value:.2f}', ha='center', va='bottom')

pdf.savefig()
plt.close()

metrics_df = pd.DataFrame(metrics)
metrics_df.plot(kind='bar', figsize=(14, 8), colormap='viridis')
```

```
plt.title('Detailed Model Performance Metrics')
plt.ylim(0, 1)
plt.ylabel('Score')
plt.xticks(rotation=0)
pdf.savefig()
plt.close()


pdf.close()
print(f'Model evaluation report saved to {pdf_path}')
```

Trained Model:

The file of Trained Model is in the zip-folder provided.