The Hong Kong Polytechnic University

Department of Electrical and Electronics Engineering

EIE4430 Honours Project

2024-2025 Semester 1

Student Name: Chan Hou Ting Constant (21034774d)

Project Title: **Machine learning model to predict the risk of diabetes**

Progress Report (1/10/2024)

Works did in past month

I tried to implement Logistic Regression and XGBoost classifier to test the performance of the model. For the Logistic Regression, I applied K-fold cross-validation to ensure the train data and test data could be fully utilized and learned. Moreover, I used GridSearch function to get the best parameters by setting a grid of parameters in both models. During the tuning in XGBoost, I found that the accuracy is the same but the recall is decreased after tuned. Maybe needs to adjust the hyperparameters to get a better performance.

Logistic Regression

```
#Logistic Regression

from sklearn.linear_model import LogisticRegression
k = 10
kf = KFold(n_splits=k, random_state=None)
acc_score = []


LogReg= LogisticRegression(max_iter=200000)


for train_index , test_index in kf.split(X):

    LogReg.fit(X_train,y_train)
    pred_LogReg = LogReg.predict(X_test)

    acc = f1_score(y_test, pred_LogReg)
    acc_score.append(acc)

avg_acc_score = sum(acc_score)/k

print('F1 score of each fold - {}'.format(acc_score))
print('Avg F1 score : {}'.format(avg_acc_score))


print(classification_report(y_test,pred_LogReg))
print(confusion_matrix(y_test,pred_LogReg))
```

```
F1 score of each fold - [0.616822429906542, 0.616822429906542, 0.616822429906542, 0.616822429906542, 0.616822429906542, 0.616822429906542, 0.616822429906
542, 0.616822429906542, 0.616822429906542, 0.616822429906542]
Avg F1 score : 0.6168224299065421
              precision    recall  f1-score   support

           0       0.75      0.84      0.80        95
           1       0.69      0.56      0.62        59

    accuracy                           0.73       154
   macro avg       0.72      0.70      0.71       154
weighted avg       0.73      0.73      0.73       154

[[80 15]
 [26 33]]
```

XGBoost (Before Tunning)

```
[122]: #XGBoost Classifier (Before Tunning)
from xgboost import XGBClassifier
XGB = XGBClassifier().fit(X_train,y_train)
#XGB.fit(X_train,y_train)
pred_XGB = XGB.predict(X_test)
print(classification_report(y_test,pred_XGB))
print(confusion_matrix(y_test,pred_XGB))
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78        95
           1       0.65      0.63      0.64        59

    accuracy                           0.73       154
   macro avg       0.71      0.71      0.71       154
weighted avg       0.73      0.73      0.73       154

[[75 20]
 [22 37]]
```

XGBoost (GridSearch function)

```
#XGBoost Classifier (Hyperparameter tunning)

param_grid = {
    'n_estimators': [100, 200, 400, 600, 800, 1000],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [1, 2, 4, 8, 10, 16, 20],
    'colsample_bytree': [0.3, 0.4, 0.5, 0.6, 0.7],
}

XGB = XGBClassifier(random_state=42)
XGB_TuningAfter = GridSearchCV(XGB, param_grid, cv=10, scoring='accuracy')
XGB_TuningAfter.fit(X_train, y_train)


#RF_Best.fit(X_train, y_train)
print("Best parameters:", XGB_TuningAfter.best_params_)
```

```
Best parameters: {'colsample_bytree': 0.5, 'learning_rate': 0.01, 'max_depth': 2, 'n_estimators': 400}
```

XGBoost (After Tunning)

```
#XGBoost Classifier (After Tunning)
XGB_BestPara = XGBClassifier(n_estimators=600, learning_rate=0.01, max_depth=2, colsample_bytree=0.7).fit(X_train, y_train)
pred_XGB_BestPara = XGB_BestPara.predict(X_test)

# Evaluate the model on the training and validation data
XGB_train_accuracy = XGB_BestPara.score(X_train, y_train)
XGB_val_accuracy = XGB_BestPara.score(X_test, y_test)


# Print the results
print("Training Accuracy:", XGB_train_accuracy)
print("Validation Accuracy:", XGB_val_accuracy)
print(classification_report(y_test,pred_XGB_BestPara))
print(confusion_matrix(y_test,pred_XGB_BestPara))
```

```
Training Accuracy: 0.8175895765472313
Validation Accuracy: 0.7337662337662337
              precision    recall  f1-score   support

           0       0.76      0.83      0.79        95
           1       0.68      0.58      0.62        59

    accuracy                           0.73       154
   macro avg       0.72      0.70      0.71       154
weighted avg       0.73      0.73      0.73       154

[[79 16]
 [25 34]]
```