MACHINE LEARNING PROJECT

Project submitted by:

B.LOGESHWARAN

Statement: Creating a classification model to predict whether a person makes over $50k a year

CSV Data:
https://drive.google.com/file/d/12w33rMGJiFWjktXBt75EVOynPy8Q1Sgg/view?usp=drivesdk

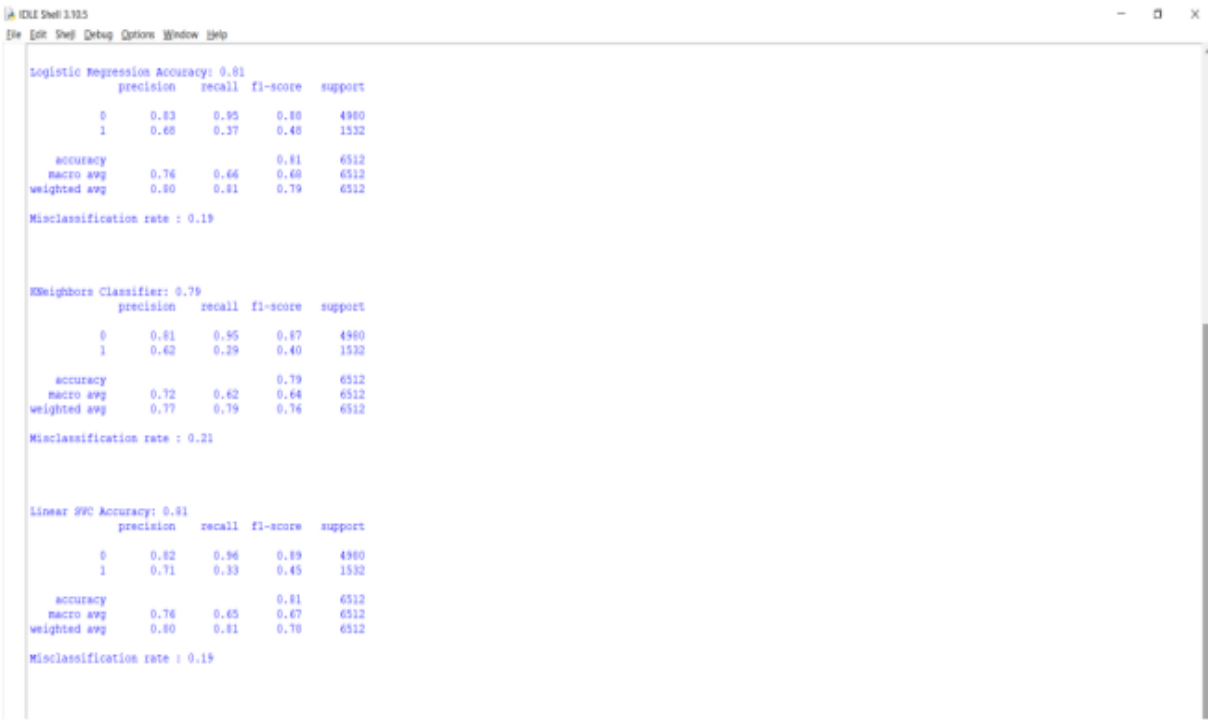
#SOURCE CODE

```
#importing essential modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVCfrom sklearn.metrics import confusion_matrix,
classification_report , ConfusionMatrixDisplay
#Reading the data and renaming the columns
my_df = pd.read_csv("C:/Users/suriy/Downloads/adult.csv")
my_df.columns=["Age","Workclass","Fnlwgt","Education","education_num","marital
_status","occupation","relationship","race","sex","capital_gain","capital_loss","hour
s_per_week","native_country","income"]
#transforming the dataframe to fitable
my_df = my_df.apply(LabelEncoder().fit_transform)
#Splitting the data
X = my_df.drop(columns = ["income"])
y = my_df["income"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

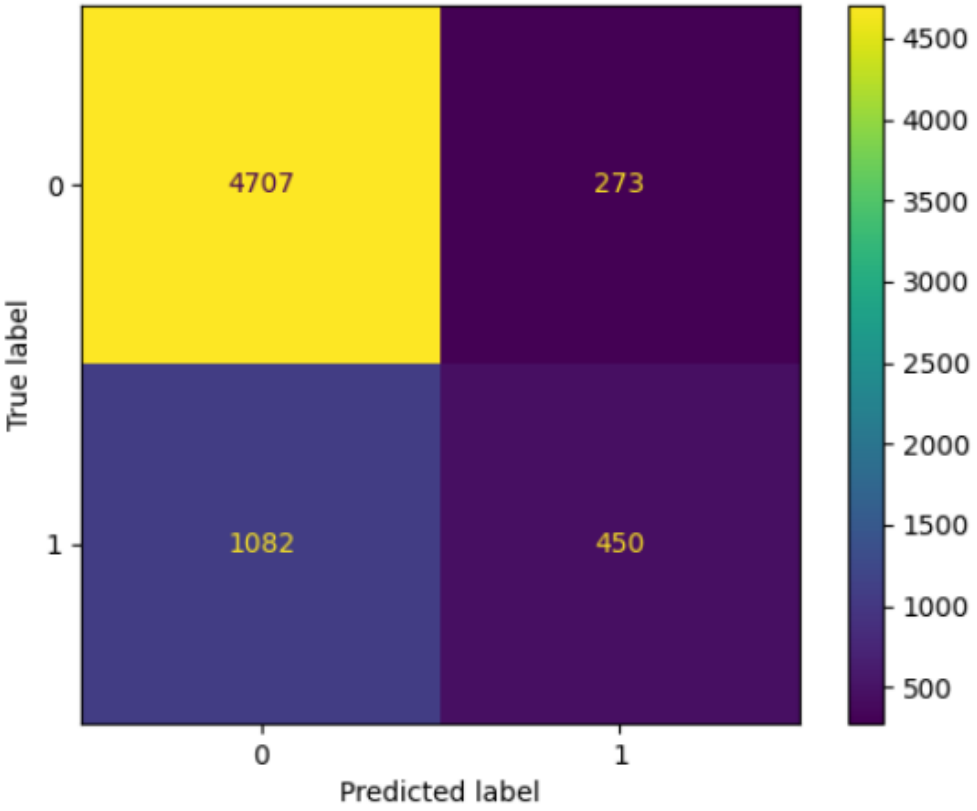
```
#Random Tree Classifier
DTC = DecisionTreeClassifier()
DTC.fit(X_train,y_train)
DTC_predict = DTC.predict(X_test)
DTC_accuracy = accuracy_score(y_test,DTC_predict)
print("Decision Tree ClassifierAccuracy: %.2f"%DTC_accuracy)#Classification report
print(classification_report(y_test, DTC_predict))
#confusion matrix
cm = confusion_matrix(y_test, DTC_predict, labels=DTC.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=DTC.classes_)
disp.plot()
plt.show()
FP = cm.sum(axis=0) - np.diag(cm)
print("Misclassification rate : %.2f\n\n\n\n"%(sum(FP)/sum(sum(cm))))
#Random Forest Classifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
RFC_predict = RFC.predict(X_test)
RFC_accuracy = accuracy_score(y_test,RFC_predict)
print("Random Forest Classifier Accuracy: %.2f"%RFC_accuracy)
#Classification report
print(classification_report(y_test, RFC_predict))
#confusion matrix
cm = confusion_matrix(y_test, RFC_predict, labels=RFC.classes_)disp =
ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=RFC.classes_)
disp.plot()
plt.show()
FP = cm.sum(axis=0) - np.diag(cm)
print("Misclassification rate : %.2f\n\n\n\n"%(sum(FP)/sum(sum(cm))))
#Logistic Regression
LGR = LogisticRegression(max_iter=3000)
LGR.fit(X_train,y_train)
LGR_predict = LGR.predict(X_test)
LGR_accuracy = accuracy_score(y_test,LGR_predict)
print("Logistic Regression Accuracy: %.2f"%LGR_accuracy)
#Classification report
print(classification_report(y_test, LGR_predict))
#confusion matrix
cm = confusion_matrix(y_test, LGR_predict, labels=LGR.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=LGR.classes_)
disp.plot()
plt.show()
FP = cm.sum(axis=0) - np.diag(cm)
print("Misclassification rate : %.2f\n\n\n\n"%(sum(FP)/sum(sum(cm))))#KNN
Classifier
KNN = KNeighborsClassifier(n_neighbors = 4)
KNN.fit(X_train,y_train)
```

```
KNN_predict = KNN.predict(X_test)
KNN_accuracy = accuracy_score(y_test,KNN_predict)
print("KNeighbors Classifier: %.2f"%KNN_accuracy)
#Classification report
print(classification_report(y_test, KNN_predict))
#confusion matrix
cm = confusion_matrix(y_test, KNN_predict, labels=KNN.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=KNN.classes_)
disp.plot()
plt.show()
FP = cm.sum(axis=0) - np.diag(cm)
print("Misclassification rate : %.2f\n\n\n\n"%(sum(FP)/sum(sum(cm))))
#Linear SVC
lsvc = LinearSVC(verbose=0,dual=False)
lsvc.fit(X_train, y_train)
lsvc_predict = lsvc.predict(X_test)
lsvc_accuracy = accuracy_score(y_test,lsvc_predict)
print("Linear SVC Accuracy: %.2f"%lsvc_accuracy)
#Classification reportprint(classification_report(y_test, lsvc_predict))
#confusion matrix
cm = confusion_matrix(y_test, lsvc_predict, labels=lsvc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=lsvc.classes_)
disp.plot()
plt.show()
FP = cm.sum(axis=0) - np.diag(cm)
print("Misclassification rate : %.2f\n\n\n\n"%(sum(FP)/sum(sum(cm))))
```

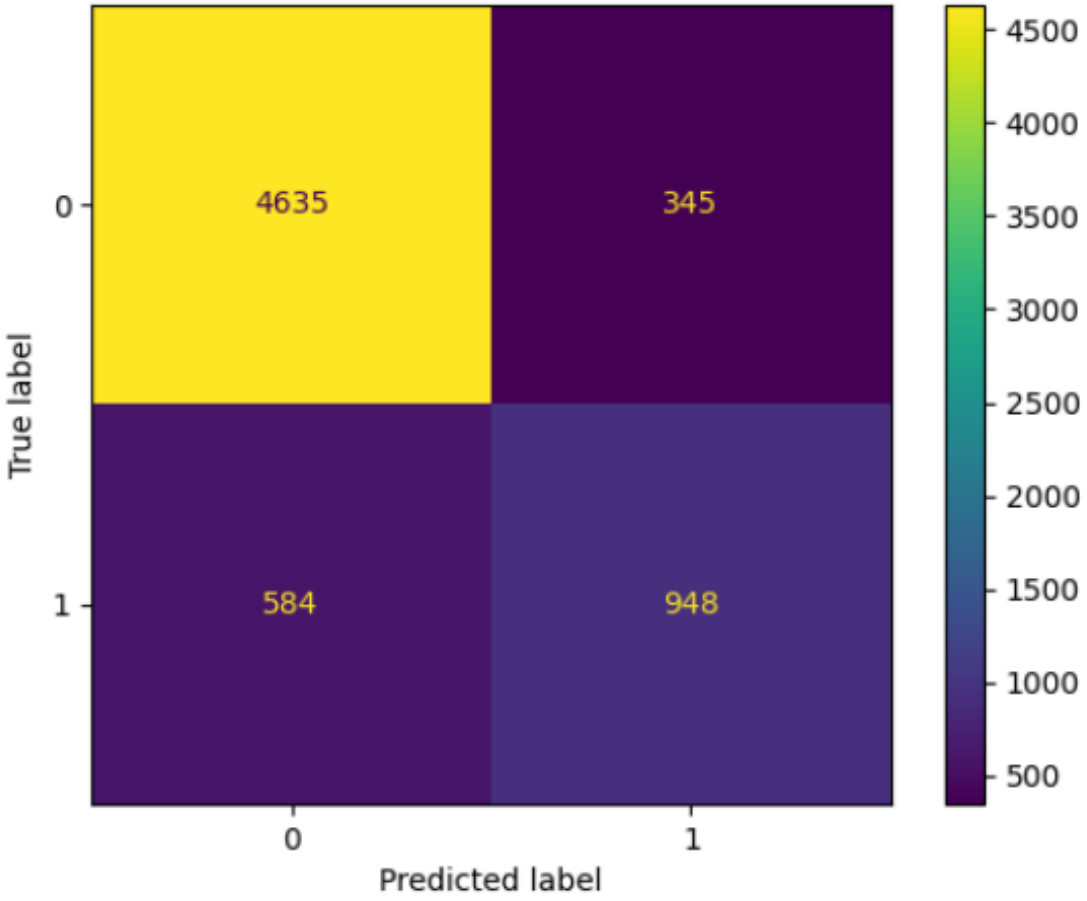
OUTPUT


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Logistic Regression Accuracy: 0.81
precision recall f1-score support
0 0.83 0.95 0.88 4980
1 0.68 0.37 0.48 1532
accuracy 0.81 6512
macro avg 0.76 0.66 0.68 6512
weighted avg 0.80 0.81 0.79 6512
Misclassification rate : 0.19
KNeighbors Classifier: 0.79
precision recall f1-score support
0 0.81 0.95 0.87 4980
1 0.62 0.29 0.40 1532
accuracy 0.79 6512
macro avg 0.72 0.62 0.64 6512
weighted avg 0.77 0.79 0.76 6512
Misclassification rate : 0.21
Linear SVC Accuracy: 0.81
precision recall f1-score support
0 0.82 0.96 0.89 4980
1 0.71 0.33 0.45 1532
accuracy 0.81 6512
macro avg 0.76 0.65 0.67 6512
weighted avg 0.80 0.81 0.78 6512
Misclassification rate : 0.19

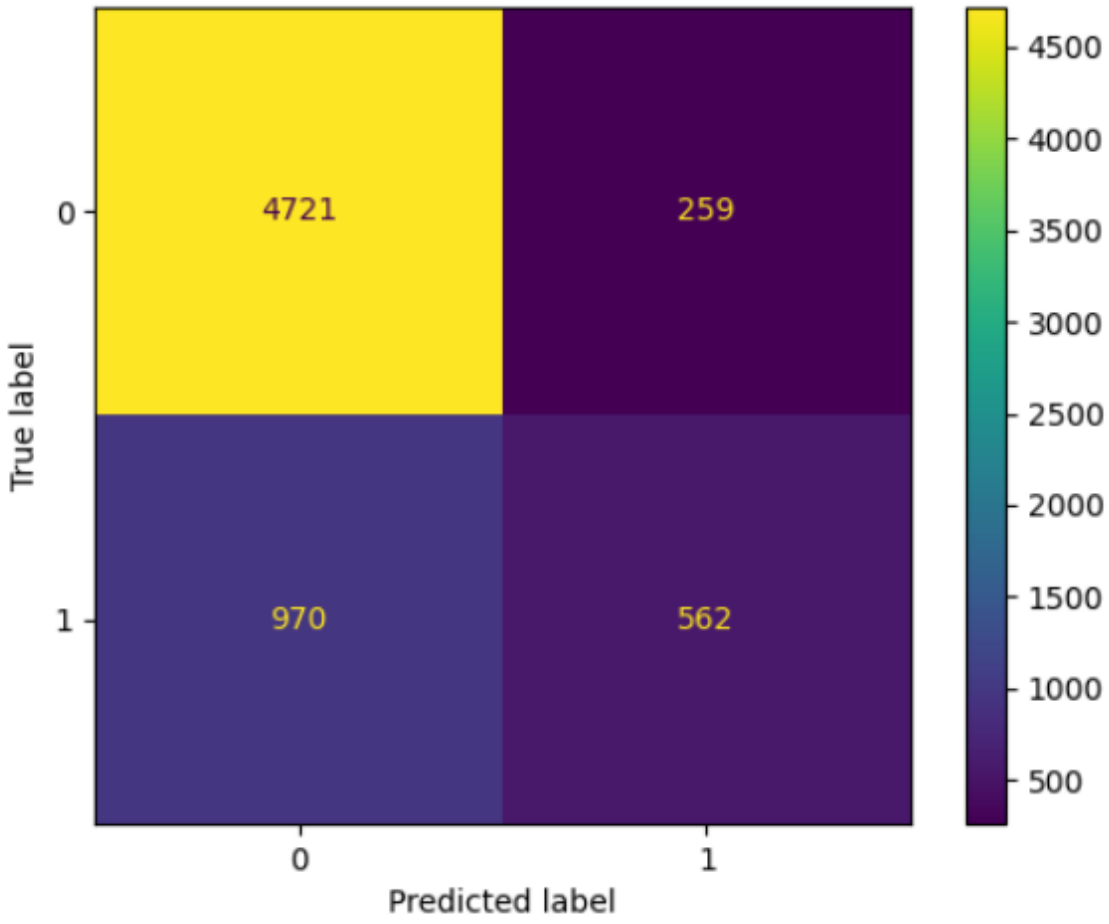
KNeighbor Classifier



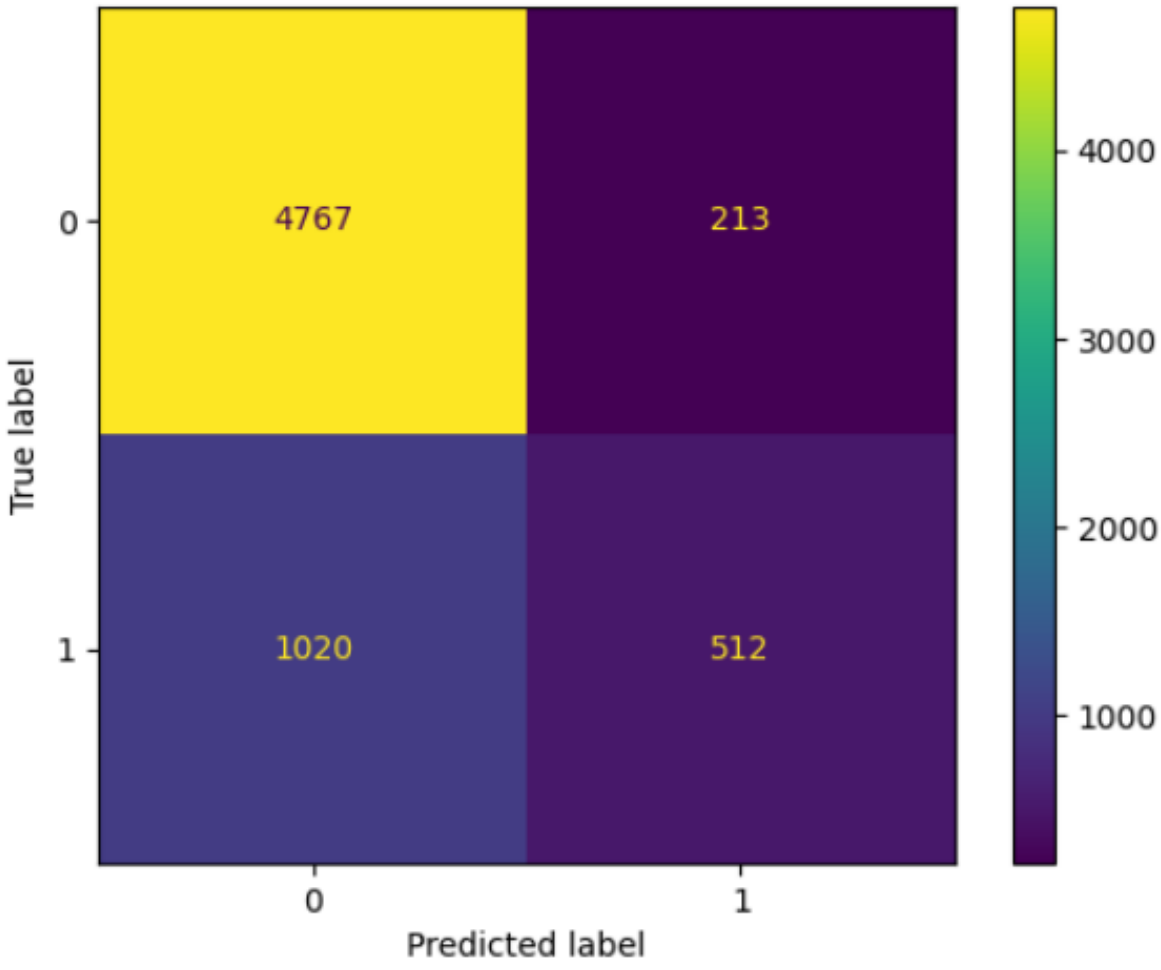
Random Forest Classifier



Logistic Regression



Linear SVC


4767
213
1020
512
0
1
True label
0
1
Predicted label
4000
3000
2000
1000