**Python Lab for Fundamental Machine Learning Techniques for Classification**

**1. Binary Logistic Regression (Binary Classification Using GLM with Different Link Functions)**

import pandas as pd

import numpy as np

from statsmodels.formula.api import glm

from statsmodels.genmod.families import Binomial

from statsmodels.genmod.families.links import logit, probit, cloglog

from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

**Dataset**

bkdt = pd.read\_excel("../input/fake-rev/simdata1.xlsx")

**Train-test split (75% train)**

np.random.seed(123)

smp\_size = int(0.75 \* len(bkdt))

train\_ind = np.random.choice(bkdt.index, size=smp\_size, replace=False)

Train = bkdt.loc[train\_ind]

Test = bkdt.drop(train\_ind)

**Train GLM models**

mdl\_logit = glm("LBL ~ RDLTY\_SR", data=Train, family=Binomial(link=logit())).fit()

mdl\_probit = glm("LBL ~ RDLTY\_SR", data=Train, family=Binomial(link=probit())).fit()

mdl\_cloglog = glm("LBL ~ RDLTY\_SR", data=Train, family=Binomial(link=cloglog())).fit()

**Show AIC of logit model**

print(f"AIC (Logit): {mdl\_logit.aic:.4f}")

**Summary**

print("\nModel Summary (Logit):")

print(mdl\_logit.summary())

**Predict probabilities**

predicted\_probs = mdl\_logit.predict(Test[['RDLTY\_SR']])

predicted\_classes = (predicted\_probs > 0.5).astype(int)

**Confusion Matrix**

cm = confusion\_matrix(Test['LBL'], predicted\_classes)

cm\_df = pd.DataFrame(cm, index=['Actual 0', 'Actual 1'], columns=['Predicted 0', 'Predicted 1'])

print("\nConfusion Matrix:\n", cm\_df)

**ROC Curve**

fpr, tpr, \_ = roc\_curve(Test['LBL'], predicted\_probs)

roc\_auc = roc\_auc\_score(Test['LBL'], predicted\_probs)

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='navy')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc='lower right')

plt.show()

**2. Multinomial Logistic Regression (More than 2 classes)(Multinomial Logistic Regression for Mobile Plan Prediction)**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

**Datase**t

data = pd.read\_csv('../input/mobile-plan/plan.csv')

**Define features and target**

X = data[['CSS', 'Duration', 'MUsage', 'Age']]

y = data['Plan']

**Train multinomial logistic regression**

model = LogisticRegression(multi\_class='multinomial', solver='lbfgs', max\_iter=1000)

model.fit(X, y)

**Coefficients**

print("Model Coefficients:")

print(model.coef\_)

**Plotting coefficients**

features = X.columns

class\_names = model.classes\_

coefficients = model.coef\_

plt.figure(figsize=(10, 6))

for i, class\_coeff in enumerate(coefficients):

plt.plot(features, class\_coeff, marker='o', label=class\_names[i])

plt.title('Logistic Regression Coefficients by Class')

plt.xlabel('Features')

plt.ylabel('Coefficient Value')

plt.axhline(0, color='grey', linewidth=0.5)

plt.legend(title='Class')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**Predicted probabilities**

probabilities = model.predict\_proba(X)

prob\_df = pd.DataFrame(probabilities, columns=model.classes\_)

print("\nPredicted Probabilities (first 5 rows):")

print(prob\_df.head())

**3. Naive Bayes and LDA for Plan Classification**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.naive\_bayes import GaussianNB

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

**Dataset**

data = pd.read\_csv('/content/plan.csv')

**Naive Bayes for Multiclass Plan Prediction**

X\_nb = data[['Age', 'MUsage', 'CSS', 'Duration']]

y\_nb = data['Plan']

X\_train\_nb, X\_test\_nb, y\_train\_nb, y\_test\_nb = train\_test\_split(

X\_nb, y\_nb, test\_size=0.25, random\_state=42, stratify=y\_nb

)

nb\_model = GaussianNB()

nb\_model.fit(X\_train\_nb, y\_train\_nb)

y\_pred\_nb = nb\_model.predict(X\_test\_nb)

y\_prob\_nb = nb\_model.predict\_proba(X\_test\_nb)

print("Naive Bayes - Classification Report:")

print(classification\_report(y\_test\_nb, y\_pred\_nb))

cm\_nb = confusion\_matrix(y\_test\_nb, y\_pred\_nb, labels=nb\_model.classes\_)

print("\nConfusion Matrix:\n", pd.DataFrame(cm\_nb,

index=[f"Actual {c}" for c in nb\_model.classes\_],

columns=[f"Predicted {c}" for c in nb\_model.classes\_]))

print("\nPredicted Probabilities (first 5 rows):")

print(pd.DataFrame(y\_prob\_nb, columns=nb\_model.classes\_).head())

**LDA for Binary Plan Classification**

binary\_data = data[data['Plan'].isin(['Vanilla', 'Pluto'])]

X\_lda = binary\_data[['MUsage', 'Age']]

y\_lda = binary\_data['Plan']

X\_train\_lda, X\_test\_lda, y\_train\_lda, y\_test\_lda = train\_test\_split(

X\_lda, y\_lda, test\_size=0.25, random\_state=42, stratify=y\_lda

)

lda\_model = LinearDiscriminantAnalysis()

lda\_model.fit(X\_train\_lda, y\_train\_lda)

y\_pred\_lda = lda\_model.predict(X\_test\_lda)

y\_prob\_lda = lda\_model.predict\_proba(X\_test\_lda)[:, 1]

print("\nLDA - Classification Report:")

print(classification\_report(y\_test\_lda, y\_pred\_lda, target\_names=lda\_model.classes\_))

cm\_lda = confusion\_matrix(y\_test\_lda, y\_pred\_lda, labels=lda\_model.classes\_)

print("\nConfusion Matrix:\n", pd.DataFrame(cm\_lda,

index=[f"Actual {c}" for c in lda\_model.classes\_],

columns=[f"Predicted {c}" for c in lda\_model.classes\_]))

print("\nLDA Coefficients (LD1):")

print(pd.DataFrame(lda\_model.coef\_.T, index=X\_lda.columns, columns=['LD1']))

print("\nClass Means:")

print(pd.DataFrame(lda\_model.means\_, index=lda\_model.classes\_, columns=X\_lda.columns))