```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, roc_curve, auc


# Load the data
tic_data = pd.read_table('2023-08-18-IS-tic.csv', sep=',', header=None)
mix_data = pd.read_csv('2023-08-18-MixDat.csv')


# Select one-tenth of the rows from both classes for the subset
subset_tic_data = pd.concat([tic_data[tic_data.index % 10 == i] for i in range(10)])
subset_mix_data = pd.concat([mix_data[mix_data.index % 10 == i] for i in range(10)])


# Split the data into training and test sets
tic_train, tic_test, mix_train, mix_test = train_test_split(subset_tic_data, subset_mix_data, test_size=0.2, random_state=42)



# Extract labels from mix_train and mix_test
y_train = mix_train['class']
y_test = mix_test['class']


# Scale and center the columns using the mean and standard deviation from the training set
scaler = StandardScaler()


# Fit the scaler on the training data and transform both training and test data
tic_train_scaled = scaler.fit_transform(tic_train)
tic_test_scaled = scaler.transform(tic_test)


# Fit the scaler on the training data and transform both training and test data
tic_train_scaled = scaler.fit_transform(tic_train)
tic_test_scaled = scaler.transform(tic_test)


# Convert the scaled data back to pandas DataFrame
tic_train_scaled_df = pd.DataFrame(tic_train_scaled, columns=tic_train.columns)
tic_test_scaled_df = pd.DataFrame(tic_test_scaled, columns=tic_test.columns)
```

Logistic Regression

```
Logit = LogisticRegressionCV(cv=10, scoring='accuracy', n_jobs=-1, max_iter = 100)


Logit.fit(tic_train_scaled_df, np.ravel(y_train))
y_pred = Logit.predict(tic_test_scaled_df)


pd.DataFrame(data = Logit.coef_, columns = tic_data.columns)
```

	0	1	2	3	4	5	6	7	8	9	...	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
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	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 2800 columns

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names="class"))


from sklearn import metrics
```

```
fpr, tpr, _ = metrics.roc_curve(y_test, Logit.predict_proba(tic_test_scaled_df))
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, y_pred)
```

```
    0.9961
```

```
auc_logit = pd.DataFrame(data = {"fpr":fpr,"tpr":tpr})
```

```
plt.figure(figsize = (15,10))
plot = sns.lineplot(x='fpr', y='tpr', data=auc_logit).set(title='AUC ROC LOGIT')
```

LDA

```
LDA = LinearDiscriminantAnalysis()
```

```
LDA.fit(tic_train_scaled_df, np.ravel(y_train))
y_pred = LDA.predict(tic_test_scaled_df)
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names='class'))
```

```
fpr, tpr, _ = metrics.roc_curve(y_test,  LDA.predict_proba(tic_test_scaled_df)[:,1])
```

```
metrics.auc(fpr, tpr)
```

```
accuracy_score(y_test, y_pred)
```

```
    0.9158
```

```
auc_LDA = pd.DataFrame(data = {"fpr":fpr,"tpr":tpr})
```

```
plt.figure(figsize = (15,10))
plot = sns.lineplot(x='fpr', y='tpr', data=auc_LDA).set(title='AUC ROC LDA')
```

KNN

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

scoring = {"AUC": "roc_auc", "Accuracy": make_scorer(accuracy_score)}

gs = GridSearchCV(
    KNeighborsClassifier(),
    param_grid={"n_neighbors": range(1, 21), "weights":["uniform","distance"], "p":[1, 2]},
    scoring=scoring,
    refit="AUC",
    return_train_score=True,
    n_jobs = -1,
    cv = 10,
    verbose = 3
)
gs.fit(tic_train_scaled_df, np.ravel(y_train))
results = gs.cv_results_
```

```
gs.best_params_
```

```
KNN = KNeighborsClassifier(n_neighbors=20, p = 2, weights = "distance")
KNN.fit(tic_train_scaled_df, np.ravel(y_train))
y_pred = KNN.predict(tic_test_scaled_df)
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=Class))
```

```
[[3112 1894]
 [   9 4985]]
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[75], line 2
      1 print(confusion_matrix(y_test, y_pred))
----> 2 print(classification_report(y_test, y_pred, target_names=Class))

NameError: name 'Class' is not defined
```

SEARCH STACK OVERFLOW

```
fpr, tpr, _ = metrics.roc_curve(y_test,  KNN.predict_proba(tic_test_scaled_df)[:,1])
```

```
metrics.auc(fpr, tpr)
```

```
accuracy_score(y_test, y_pred)
```

```
0.8097
```

```
auc_KNN = pd.DataFrame(data = {"fpr":fpr,"tpr":tpr})
```

```
plt.figure(figsize = (15,10))
plot = sns.lineplot(x='fpr', y='tpr', data=auc_KNN).set(title='AUC ROC KNN')
```

Linear SVM

```
C_range = np.logspace(-2, 10, 13)
param_grid = dict(C=C_range)
```

```
from sklearn.model_selection import StratifiedShuffleSplit
```

```
cv = StratifiedShuffleSplit(n_splits=30, test_size=0.2, random_state=101)
```

```
grid = GridSearchCV(SVC(kernel = "linear", max_iter=10000, probability=True), param_grid=param_grid, cv=cv, n_jobs = -1, verbose
```

```
grid.fit(tic_train_scaled_df, np.ravel(y_train))
```

```
results = grid.cv_results_
```

```
LSVC = SVC(kernel = "linear", max_iter=100, C = 0.1, probability = True)
LSVC.fit(tic_train_scaled_df, np.ravel(y_train))
y_pred = LSVC.predict(tic_test_scaled_df)
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=Class))
```

```
fpr, tpr, _ = metrics.roc_curve(y_test,  LSVC.predict_proba(X_test)[:,1])
```

```
metrics.auc(fpr, tpr)
```

```
accuracy_score(y_test, y_pred)
```

```
auc_LSVC = pd.DataFrame(data = {"fpr":fpr,"tpr":tpr})
```

```
plt.figure(figsize = (15,10))
plot = sns.lineplot(x='fpr', y='tpr', data=auc_LSVC).set(title='AUC ROC LINEAR SVC')
```

Radial SVC

```
gamma_range = np.logspace(-9, 3, 13)
param_grid = dict(gamma=gamma_range, C=C_range)
```

```
cv = StratifiedShuffleSplit(n_splits=3, test_size=0.2, random_state=101)
grid = GridSearchCV(SVC(probability = True), param_grid=param_grid, cv=cv, n_jobs = -1, verbose = 3)


grid.fit(X_train, np.ravel(y_train))


results = grid.cv_results_


grid.best_params_


RSVC = SVC(C = 100, gamma = 0.1, probability = True)
RSVC.fit(X_train, np.ravel(y_train))
y_pred = RSVC.predict(X_test)


print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=Class))


fpr, tpr, _ = metrics.roc_curve(y_test,  RSVC.predict_proba(X_test)[:,1])


metrics.auc(fpr, tpr)


accuracy_score(y_test, y_pred)


auc_RSVC = pd.DataFrame(data = {"fpr":fpr,"tpr":tpr})


plt.figure(figsize = (15,10))
plot = sns.lineplot(x='fpr', y='tpr', data=auc_RSVC).set(title='AUC ROC RADIAL SVC')
```