Developing a Prediction Model
Objective: *Use logistic regression (and other models) to predict patient outcomes.*

Modeling: Linear Classifiers

```
#Import packages

# Scaling
from sklearn.preprocessing import RobustScaler

# Train Test Split
from sklearn.model_selection import train_test_split

# Models
import torch
import torch.nn as nn
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Metrics
from sklearn.metrics import accuracy_score, classification_report, roc_curve

# Cross Validation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

print('Packages imported...')
```

```
Packages imported...
```

Train and test split

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)
X_train[cont_cols] =scaler.fit_transform(X_train[cont_cols])
X_test[cont_cols] =scaler.transform(X_test[cont_cols])
print("The shape of X_train is      ", X_train.shape)
print("The shape of X_test is       ",X_test.shape)
print("The shape of y_train is      ",y_train.shape)
print("The shape of y_test is       ",y_test.shape)
```

```
The shape of X_train is       (242, 22)
The shape of X_test is        (61, 22)
The shape of y_train is       (242, 1)
The shape of y_test is        (61, 1)
```

```
#Scaling and encoding features:

# creating a copy of df
df1 = df

# define the columns to be encoded and scaled
cat_cols = ['sex','exercise_induced_angina','num_major_vessels','chest_pain_type','fasting_blood_sugar','resting_ecg','slope','thal_rate']
con_cols = ["age","resting_bp","cholesterol","max_heart_rate","previous_peak"]

# encoding the categorical columns
df1 = pd.get_dummies(df1, columns = cat_cols, drop_first = True)

# defining the features and target
X = df1.drop(['risk_level'],axis=1)
y = df1[['risk_level']]
```

```
# instantiating the scaler
scaler = RobustScaler()

# scaling the continuous feature
X[cont_cols] = scaler.fit_transform(X[cont_cols])
print("The first 5 rows of X are")
X.head()
```

Linear Classifiers

Support Vector Machines

```
# instantiating the object and fitting
clf = SVC(kernel='linear', C=1, random_state=42).fit(X_train,y_train)

# predicting the values
y_pred = clf.predict(X_test)

# printing the test accuracy
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("f1 Score:", f1)
print("AUC-ROC:", auc_roc)
```

```
Accuracy: 0.8688524590163934
Precision: 0.9
Recall: 0.84375
f1 Score: 0.8709677419354839
AUC-ROC: 0.8701508620689655
```

Hyperparameter tuning of SVC

```
# instantiating the object
svm = SVC()

# setting a grid - not so extensive
parameters = {"C":np.arange(1,10,1),'gamma':[0.00001,0.00005, 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5]}

# instantiating the GridSearchCV object
searcher = GridSearchCV(svm, parameters)

# fitting the object
searcher.fit(X_train, y_train)

# the scores
print("The best params are :", searcher.best_params_)
print("The best score is   :", searcher.best_score_)

# predicting the values
y_pred = searcher.predict(X_test)

# printing the test accuracy
print("The test accuracy score of SVM after hyper-parameter tuning is ", accuracy_score(y_test, y_pred))
```

```
The best params are : {'C': 3, 'gamma': 0.1}
The best score is   : 0.8384353741496599
The test accuracy score of SVM after hyper-parameter tuning is  0.9016393442622951
```

Note: This test accuracy score, 0.90, is the highest that our models reached in this process.

Logistic Regression

```
# instantiating the object
logreg = LogisticRegression()

# fitting the object
logreg.fit(X_train, y_train)

# calculating the probabilities
y_pred_proba = logreg.predict_proba(X_test)

# finding the predicted valued
y_pred = np.argmax(y_pred_proba,axis=1)

# printing the test accuracy
print("The test accuracy score of Logistic Regression is ", accuracy_score(y_test, y_pred))
```

```
The test accuracy score of Logistic Regression is  0.9016393442622951
```

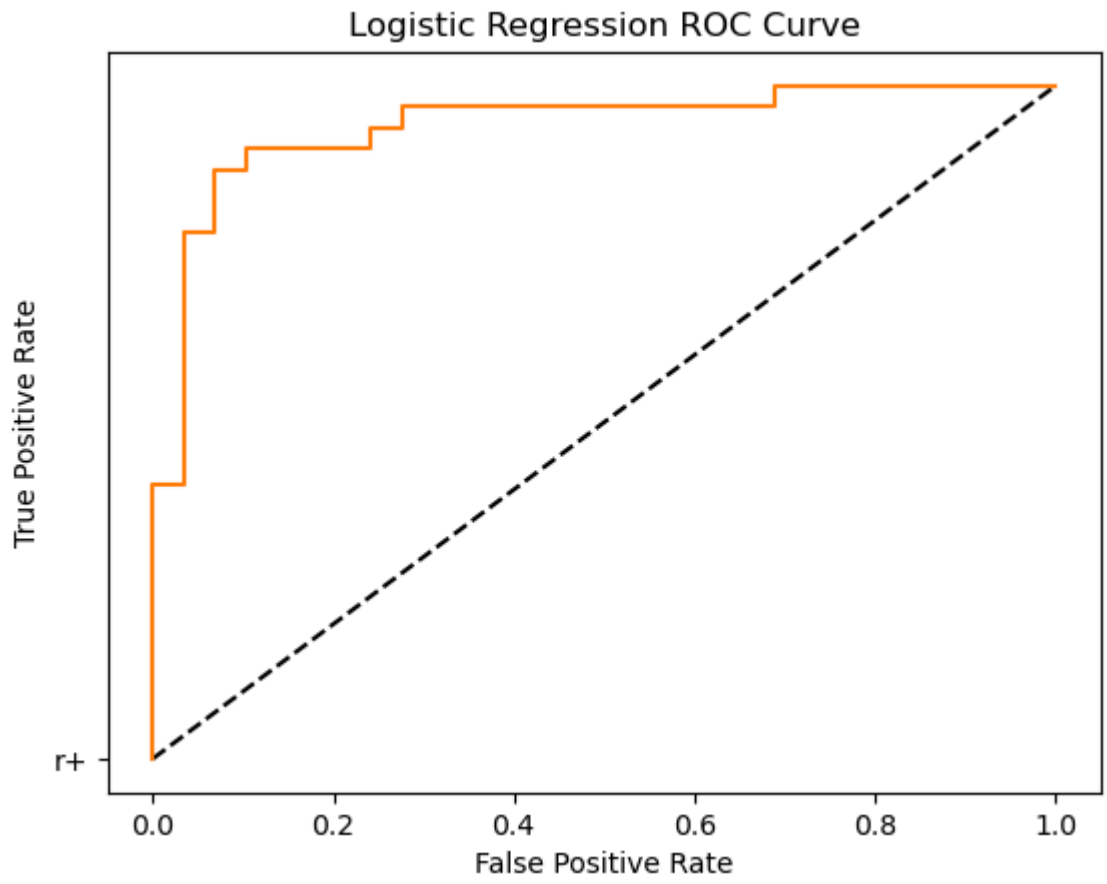
Note: This test accuracy score, 0.90, matches what was obtained by the SVM after tuning.

ROC Curve

```
# calculating the probabilities
y_pred_prob = logreg.predict_proba(X_test)[:,1]

# instantiating the roc_cruve
fpr,tpr,threshols=roc_curve(y_test,y_pred_prob)

# plotting the curve
plt.plot([0,1],[0,1],"k--",'r+')
plt.plot(fpr,tpr,label='Logistic Regression')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Regression ROC Curve")
plt.show()
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
# Assuming 'y_test' is the actual labels and 'y_pred' are the model predictions on the test set
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred_prob)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("f1 Score:", f1)
print("AUC-ROC:", auc_roc)
```

```
Accuracy: 0.9016393442622951
Precision: 0.9333333333333333
Recall: 0.875
f1 Score: 0.9032258064516129
AUC-ROC: 0.9396551724137931
```

Note: These scores for the logistic regression model seem high enough to be acceptable.

Tree Models

Decision Tree

```
# instantiating the object
dt = DecisionTreeClassifier(random_state = 42)

# fitting the model
dt.fit(X_train, y_train)

# calculating the predictions
y_pred = dt.predict(X_test)

# printing the test accuracy
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("f1 Score:", f1)
print("AUC-ROC:", auc_roc)
```

```
Accuracy: 0.7868852459016393
Precision: 0.8518518518518519
Recall: 0.71875
f1 Score: 0.7796610169491526
AUC-ROC: 0.7904094827586206
```

Random Forest

```
# instantiating the object
rf = RandomForestClassifier()

# fitting the model
rf.fit(X_train, y_train)

# calculating the predictions
y_pred = dt.predict(X_test)

# printing the test accuracy
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("f1 Score:", f1)
print("AUC-ROC:", auc_roc)
```

```
Accuracy: 0.7868852459016393
Precision: 0.8518518518518519
Recall: 0.71875
f1 Score: 0.7796610169491526
AUC-ROC: 0.7904094827586206
```

Gradient Boosting Classifier

```
# instantiate the classifier
gbt = GradientBoostingClassifier(n_estimators = 300,max_depth=1,subsample=0.8,max_features=0.2,random_state=42)

# fitting the model
gbt.fit(X_train,y_train)

# predicting values
y_pred = gbt.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("f1 Score:", f1)
print("AUC-ROC:", auc_roc)
```

```
Accuracy: 0.8688524590163934
Precision: 0.9
Recall: 0.84375
f1 Score: 0.8709677419354839
AUC-ROC: 0.8701508620689655
```

Perform Cross-Validation

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score
```

```
clf = DecisionTreeClassifier(random_state=42)
```

```
k_folds = KFold(n_splits = 10)

scores = cross_val_score(clf, X, y, cv = k_folds)
```

```
print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

```
Cross Validation Scores:  [0.58064516 0.80645161 0.67741935 0.63333333 0.73333333 0.8
 0.63333333 0.76666667 0.7        0.43333333]
Average CV Score:  0.6764516129032259
Number of CV Scores used in Average:  10
```

Conclusion: The test accuracy score, 0.90, of SVM (after hyperparameter tuning) and Logistic Regression is the best of the models fitted with all features.

In addition, the precision score of 0.93 and recall score of 0.88 are high enough for us to regard the logistic regression model as adequate.