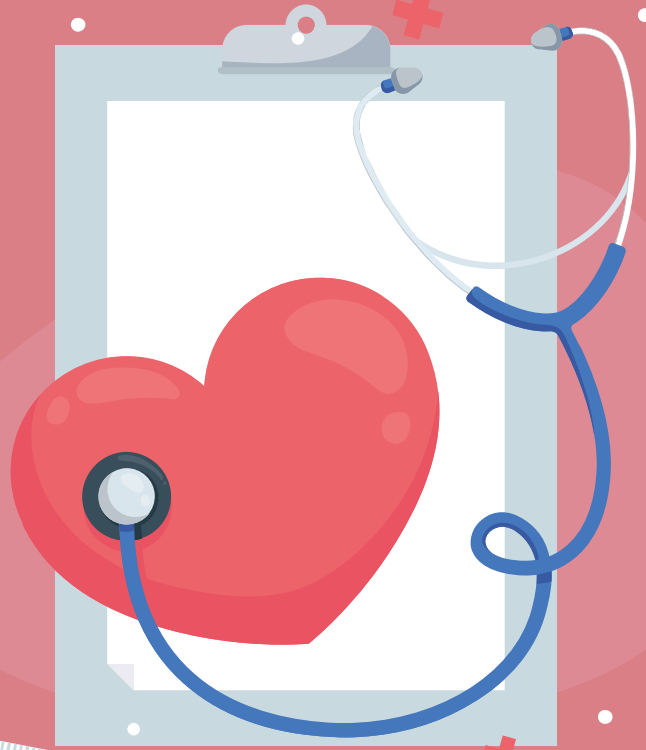
CARDIOVASCULAR DISEASE



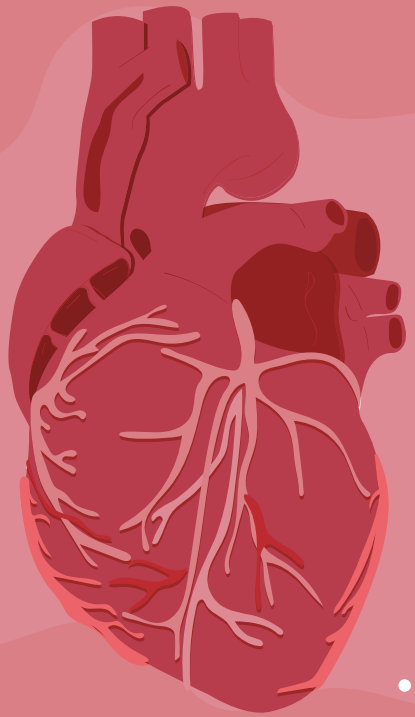
Janith Perera

PURPOSE OF THE PROJECT

- Predict a posible Cardiovascular (Heart) Disease from risk factors which can be caused heart failures
- Compare the prediction accuracy of the target variable which is Heart Disease with using following classification models
    1. Logistics Regression
    2. Decision Tree
    3. Random Forest
    4. Support Vector Machine
    5. Stochastic Gradient Descent
    6. XGBoost
- Target variable of the datast is binary variable
- 11 input variables such as Age, Sex, Chest pain type, Cholesterol, Resting ECG and etc.

SIGNIFICANCE OF THE PROJECT

- Cardiovascular diseases are the number 1 reason of death worldwide
- 4 out of 5 cardiovascular disease deaths are due to heart attacks and strokes
- It is beneficial if we can predict the heart disease accurately in order to take precautions



RESEARCH QUESTION

Are Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, Stochastic Gradient Descent, XGBoost algorithms good models for predicting the possible heart disease based on clinical risk factors like chest pain type, cholesterol level, resting ECG, fasting blood sugar, maximum heart rate and etc.

DATASET

Target variable:

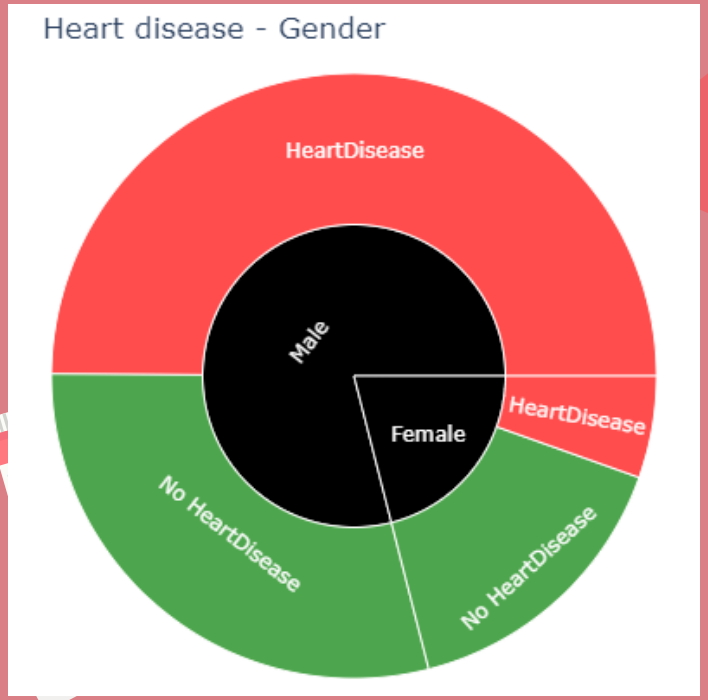
- Binary value – HeartDisease [1:HeartDisease, 0:Normal]

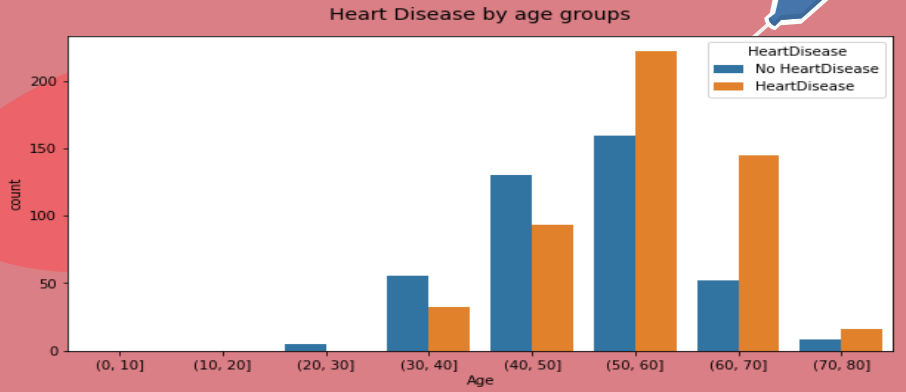
Input varables:

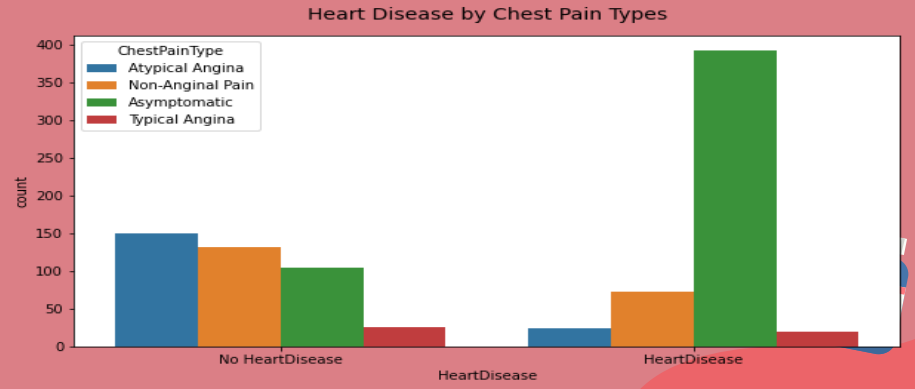
- 11 variables
- Categorical values – Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope
- Numarical values – Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak

Zero null-values
918 intences
Downloaded from Kaggle.com

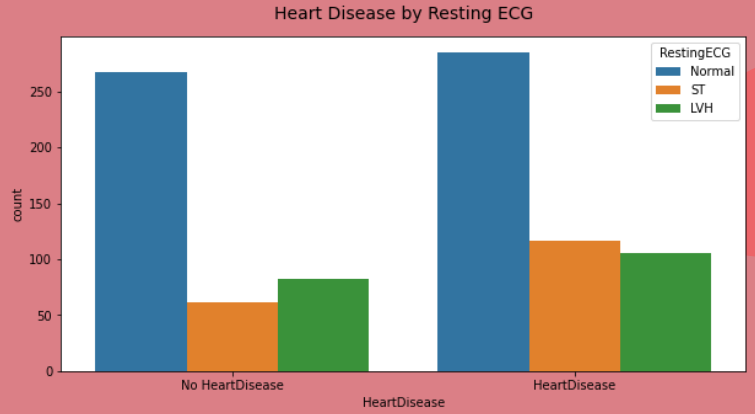
EXPLORATORY DATA ANALYSIS

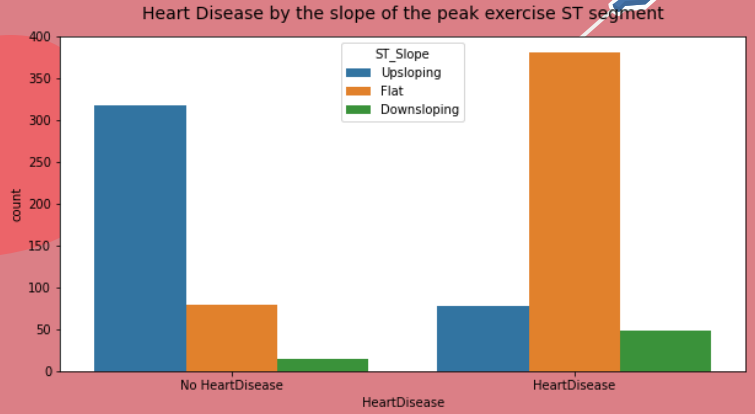


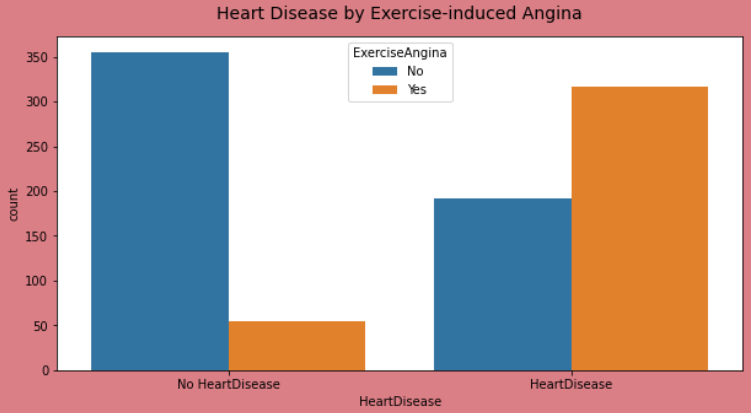




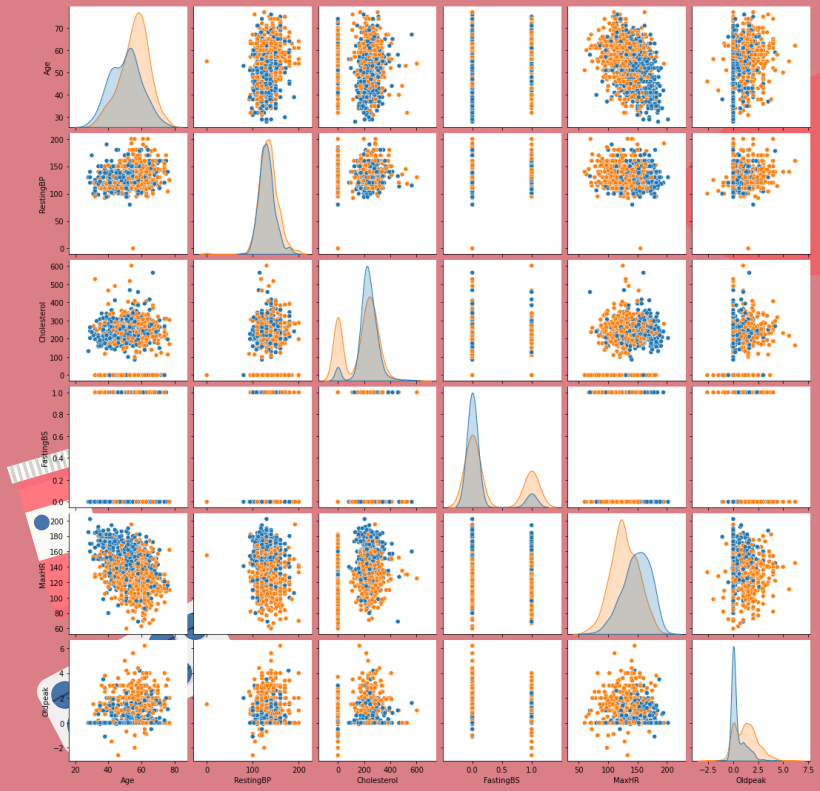
EXPLORATORY DATA ANALYSIS

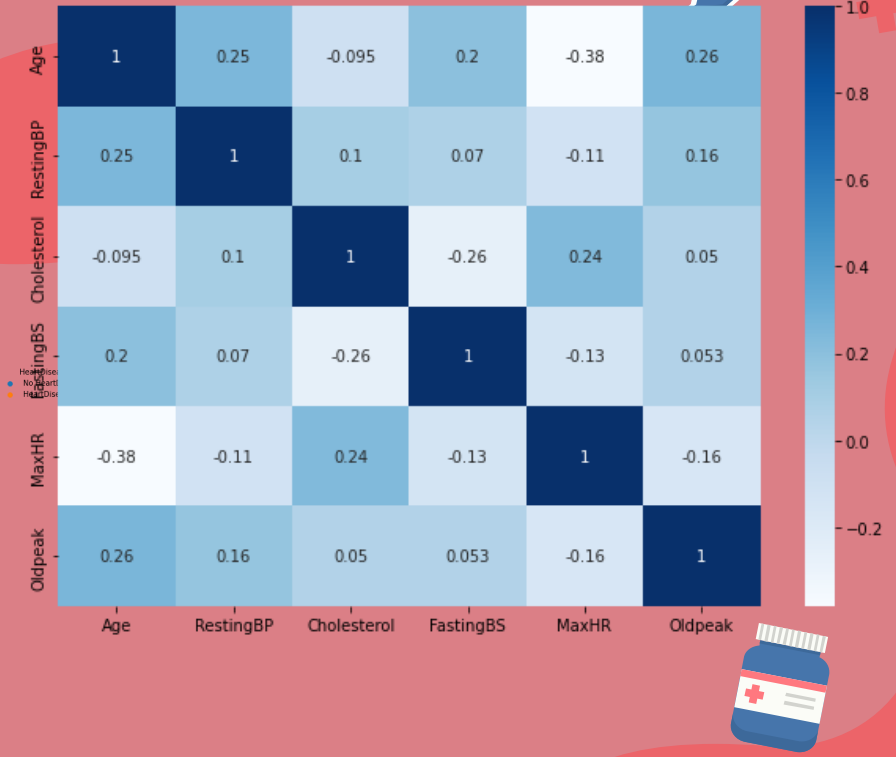






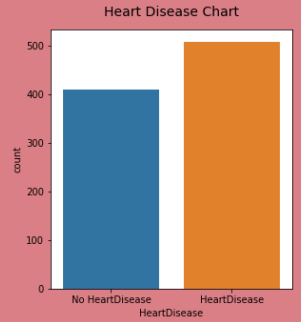
EXPLORATORY DATA ANALYSIS

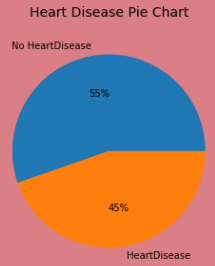




DATA SPLITTING

- Balanced target variable





- Used get_dummies() method to one-hot-encode the categorical variables
- Used StandardScaler() method to scale the numarical variables
- Used train_test_split() method to split the dataset into 70% of training and 30% testing sets

```
1  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
2  print(X_train.shape, y_train.shape)
3  print(X_test.shape, y_test.shape)
```

✓ 0.1s

```
(642, 20) (642,)
(276, 20) (276,)
```

MODEL BUILDING AND COMPARISON

- Initialy used six models clasification models
  - Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine Classifier, Stochastic Gradient Desent Classifier and XG Boost Classifier

```python
model_table = PrettyTable()
model_table.field_names = ['Model', 'Accuracy Score', 'Precision Score', 'Recall Score', 'F1 Score', 'ROC-AUC Score', 'Log-loss Score']

models = [LogisticRegression(random_state=42),
          DecisionTreeClassifier(criterion='gini', random_state=42),
          RandomForestClassifier(n_estimators=100, max_features=None, random_state=42),
          SVC(kernel='rbf', random_state=42),
          SGDClassifier(max_iter=1000, random_state=42),
          GradientBoostingClassifier(n_estimators=100, max_features=None, random_state=42)]

for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc_score = 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)
    roc_auc = roc_auc_score(y_test, y_pred)
    lg_loss = log_loss(y_test, y_pred)
    model_table.add_row([type(model).__name__, format(acc_score, '.2f'), format(precision, '.2f'), format(recall, '.2f'), format(f1, '.2f'), format(roc_auc, '.2f'), format(lg_los
print('Comparison of accuracy scores for the test set in different models')
print(model_table)
```

✓ 1.2s Python

MODEL BUILDING AND COMPARISON

Comparison of accuracy scores for the test set in different models

Model	Accuracy Score	Precision Score	Recall Score	F1 Score	ROC-AUC Score	Log-loss Score
LogisticRegression	0.88	0.92	0.88	0.90	0.89	4.00
DecisionTreeClassifier	0.78	0.88	0.73	0.79	0.79	7.76
RandomForestClassifier	0.85	0.90	0.84	0.87	0.85	5.26
SVC	0.88	0.90	0.91	0.90	0.88	4.00
SGDClassifier	0.75	0.91	0.65	0.76	0.78	8.51
GradientBoostingClassifier	0.88	0.93	0.86	0.89	0.88	4.25

- Selected best three models with highest accuracy scores and remodeled with cross validation to select optimal models
- Best three models

  Logistic Regression, Support Vector Machine Classifier and XG Boost Classifier

MODEL OPTIMIZATION

- Selected best model out of them using cross validation method

```python
cv_table = PrettyTable()
cv_table.field_names = ['Model', 'CV Score']

cv_models = [LogisticRegression(random_state=42),
             SVC(kernel='rbf', random_state=42),
             GradientBoostingClassifier(n_estimators=100, max_features=None, random_state=42)]
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=5, random_state=42)

for model in cv_models:
    scores = cross_val_score(model, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
    cv_scores = np.mean(scores)
    cv_table.add_row([type(model).__name__, format(cv_scores, '.2f')])
print('Comparison of cross validation scores for the test set in different models')
print(cv_table)
```

✓ 14.4s Python

```
Comparison of cross validation scores for the test set in different models
+---------------------------+----------+
|           Model           | CV Score |
+---------------------------+----------+
|     LogisticRegression    |   0.85   |
|            SVC            |   0.87   |
| GradientBoostingClassifier |   0.86   |
+---------------------------+----------+
```

MODEL OPTIMIZATION AND SELECTION

- Selected the Support Vector Machine Classifier as the best performer among the models
- Tuned with the hyperparameters to select the model

```python
sv_clf = SVC(random_state=42)
parameters = [{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
               'C': [1.0, 2.0, 5.0, 10.0],
               'gamma': ['scale', 'auto', 0.0001, 0.001, 0.01, 0.1, 1],
               'shrinking': [True, False],
               'probability': [True, False]}]
gs_sv = GridSearchCV(sv_clf, param_grid=parameters, cv=3)
gs_sv.fit(X_train, y_train)
best_sv = gs_sv.best_estimator_
best_sv.get_params()
```

✓ 52.4s Python

- Hyperparameter tuned model has the overall accuracy score of 0.90 on training set and 0.88 on testing set

Model Results

Classification Reports

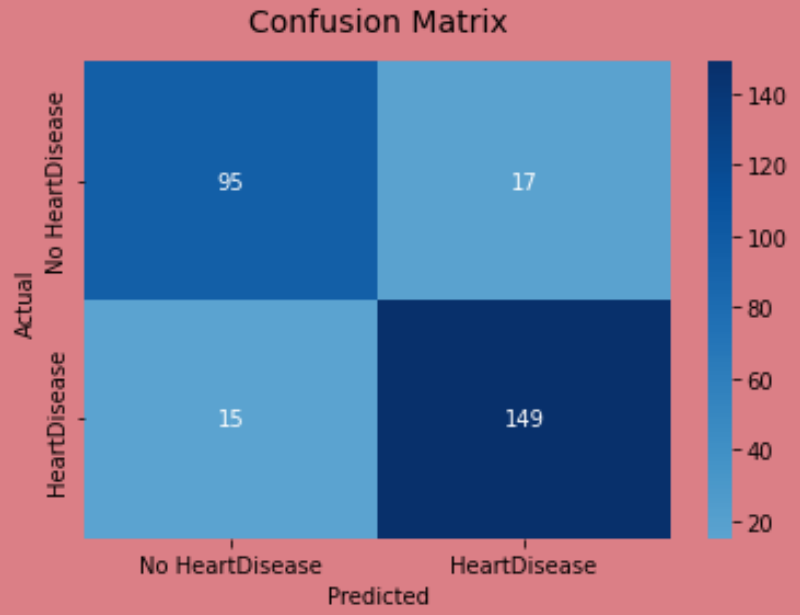
Classification report on training set

	precision	recall	f1-score	support
0	0.92	0.86	0.89	298
1	0.89	0.93	0.91	344
accuracy			0.90	642
macro avg	0.90	0.90	0.90	642
weighted avg	0.90	0.90	0.90	642

Classification report on testing set

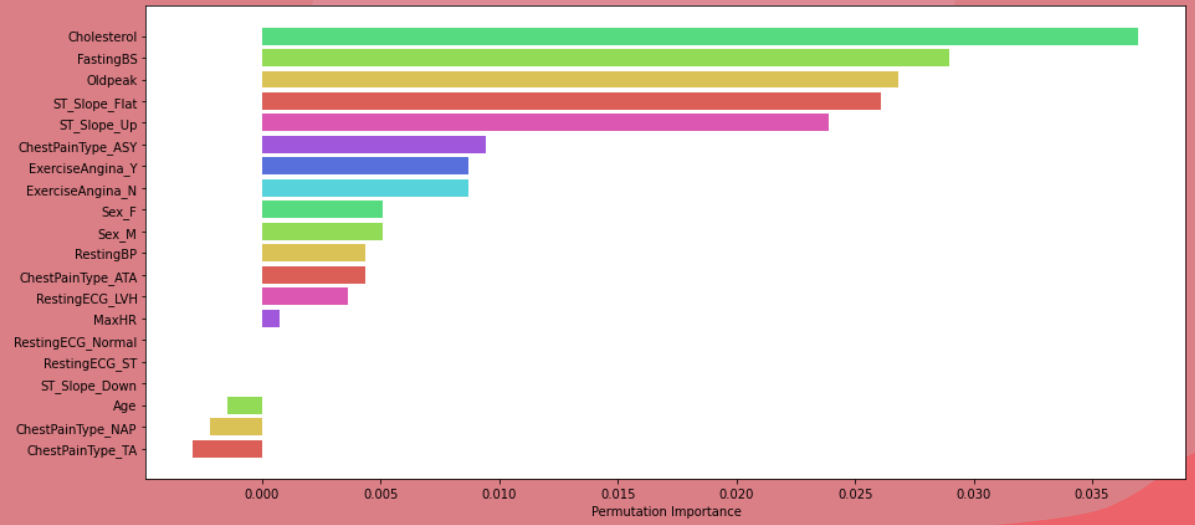
	precision	recall	f1-score	support
0	0.86	0.85	0.86	112
1	0.90	0.91	0.90	164
accuracy			0.88	276
macro avg	0.88	0.88	0.88	276
weighted avg	0.88	0.88	0.88	276

Confusion Matrix on Test set



CONCLUSION AND RECOMMENDATION

- For this dataset Support Vector Classification model worked best
- With the hyperparameter tuning model prediced the target variable with 88% accuracy
- Cholesterol, FastingBS, Oldpeak, ST_Slop_Flat, ST_Slop_UP are the most important features of predicting heart disease



THANK YOU