IBM SkillsBuild

Data Analytics Internship Programme PPT

TEAM NEON

IBM SkillsBuild

CSRBOX

CARDIOVASCULAR DISEASE

WORLD HEART FEDERATION

CARDIOVASCULAR DISEASE

THE WORLD'S NUMBER 1 KILLER

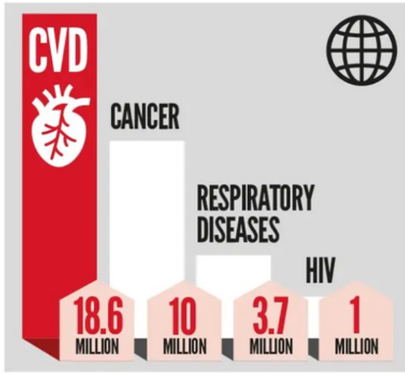
Cardiovascular diseases are a group of disorders of the heart and blood vessels, commonly referred to as **heart disease** and **stroke**.

18.6 MILLION deaths every year from CVD

33% of all global deaths

>75% of CVD deaths take place in low- and middle-income countries

GLOBAL CAUSES OF DEATH



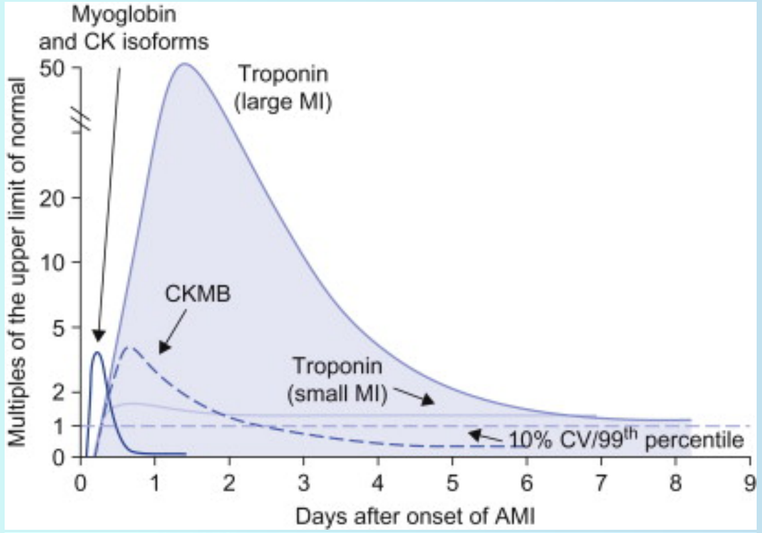
RISK FACTORS FOR CVD

- High Blood Pressure
- Unhealthy Diet
- High Cholesterol
- Diabetes
- Overweight & Obesity
- Tobacco
- Air Pollution
- Kidney Disease
- Physical Inactivity
- Harmful use of alcohol

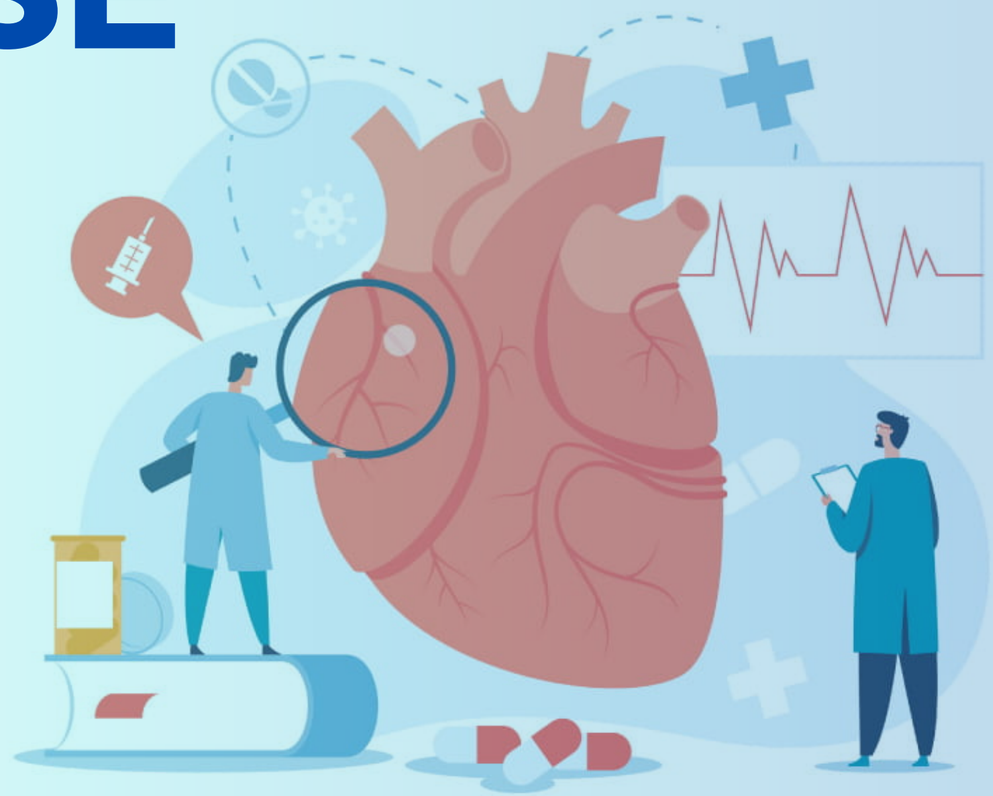
Sources: World Health Organization; IHME, Global Burden of Disease

info@worldheart.org
www.worldheart.org

worldheartfederation
worldheartfed
worldheartfederation



CARDIOVASCULAR HEART DISEASE PREDICTION



TEAM MEMBERS

P.ANUDEEP

K.S.V.RAKESH

M.RAHUL

K.KARTHIKEYA

B.NIKHITHA

K.SUMEDHA

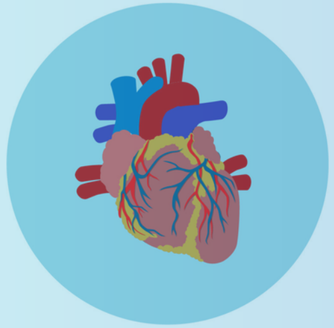
G.RISHIKA

E.SHASHANKA

T.CHINMAYEE

INTRODUCTION

- Heart disease, also known as cardiovascular disease, refers to a group of conditions that affect the heart and blood vessels. It is a major global health concern and a leading cause of mortality worldwide.
- **The effects of heart disease can be severe and life-threatening**. They can range from chest pain (angina) and shortness of breath to heart attacks and strokes. **In advanced stages, heart disease can lead to heart failure**, a condition where the heart is unable to pump blood efficiently, causing fatigue, fluid retention, and organ damage.



- Major Causes of Heart Disease worldwide:
  - High blood pressure, High cholesterol levels
  - Change in lifestyle, Change in the food habits, lack of physical activity
  - Smoking and alcohol consumption

- ***Deaths from cardiovascular disease (CVD) jumped globally from 12.1 million in 1990 to 20.5 million in 2021, according to a new report from the World Heart Federation (WHF).***

- An estimated 17.9 million people died from CVDs in 2019, representing 32% of all global deaths. Of these deaths, 85% were due to heart attack and stroke.

- More than 4 out of 5 CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age.

SUMMARY

- Indeed, the rising rate of heart disease is a global problem that has motivated researchers to aggressively look for strategies to predict it and avoid it. The use of current data and cutting-edge technologies gained prominence in this endeavor.

The project described here focuses on ***data visualization and analysis using Python libraries*** like ***NumPy, Pandas, Matplotlib, and Scikit-learn***, specifically in the context of heart disease prediction

By combining the power of Python libraries for data manipulation and visualization with ***Random Forest Classifier as the predictive model***, this project aims to contribute to the development of accurate and efficient heart disease prediction systems.

The ***ultimate goal is to facilitate early detection*** and intervention, which can significantly ***improve patient outcomes and reduce the global burden of heart disease***.

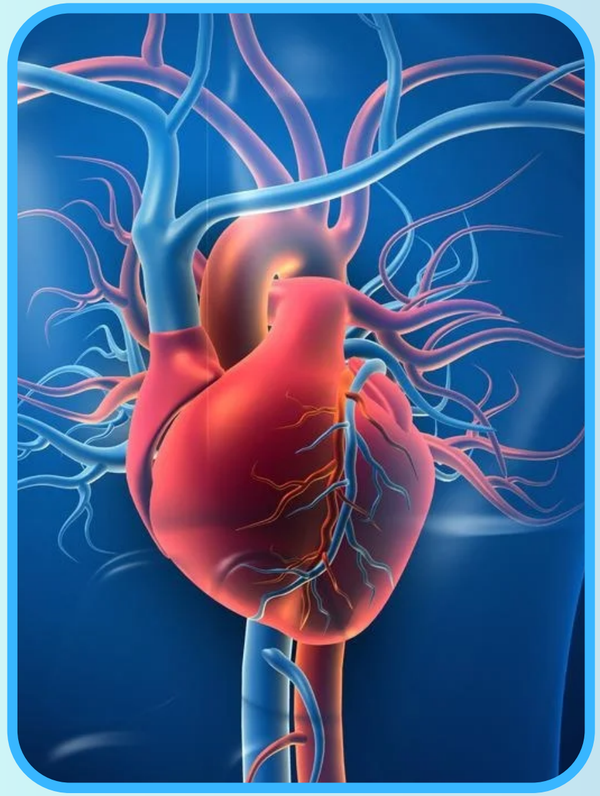
PURPOSE OF THE PROJECT

- The Cardiovascular Heart Disease Prediction project uses a dataset of 1025 samples and 14 features, some of which include age, gender, chest pain type, resting blood pressure, serum cholestoral, fasting blood sugar, maximum heart rate achieved etc

- ***Data analysis*** has been done to arrive at the number of patients affected with heart disease and what are the major attributes contributing for the same.

- The ***machine learning model*** helps us in determining whether the patient is believed to have exposed to heart disease or not.

- The ***goal of the project*** is to provide a detailed analysis and accurate prediction from the real world dataset available on kaggle.



DATASET SOURCE

Name of the Dataset:

Heart Disease Dataset

Source: Kaggle

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset?resource=download

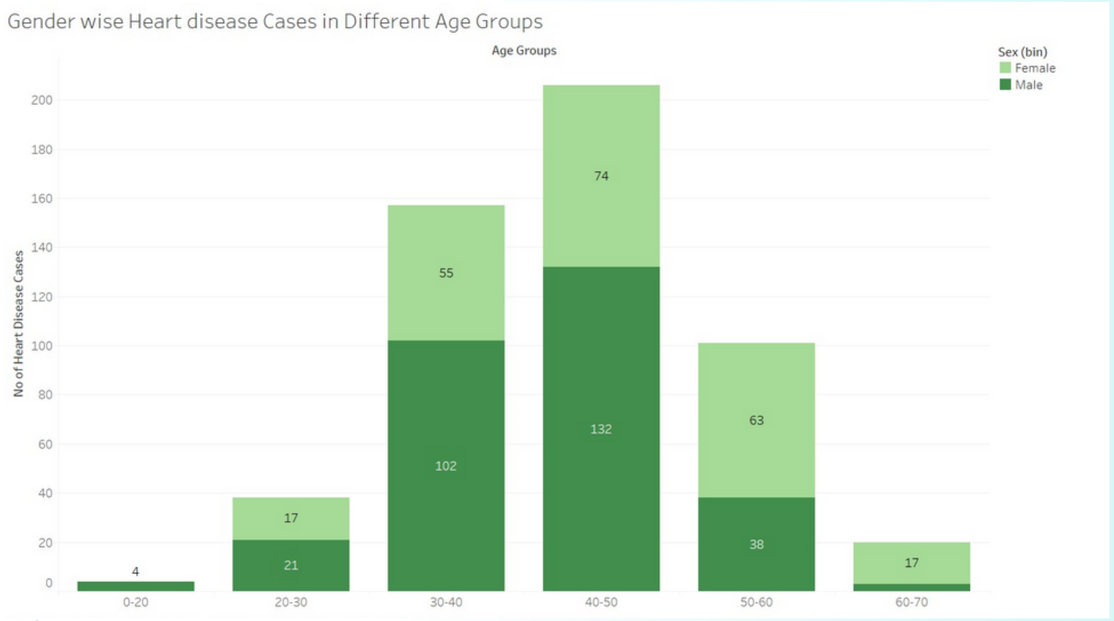
There are several attributes in the dataset, such as, age, gender, chest pain type, resting blood pressure, serum cholestoral, fasting blood sugar etc., on which we are predicting, analyzing, visualizing, and detecting the cardiovascular disease.

# age age in years	# sex (1 = male; 0 = female)	# cp chest pain type	# trestbps resting blood pressure (in mm Hg on admission to the hospital)	# chol serum cholestoral in mg/dl	# fbs (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)	# restecg resting electrocardiographic results
29 – 77	0 – 1	0 – 3	94 – 200	126 – 564	0 – 1	0 – 2
52	1	0	125	212	0	1
53	1	0	140	203	1	0
70	1	0	145	174	0	1
61	1	0	148	203	0	1
62	0	0	138	294	1	1
58	0	0	100	248	0	0
58	1	0	114	318	0	2
55	1	0	160	289	0	0
46	1	0	120	249	0	0
54	1	0	122	286	0	0
71	0	0	112	149	0	1

# thalach 	
maximum heart rate achieved
	# exang
	exercise induced angina (1 = yes; 0 = no)
		# oldpeak
		ST depression induced by exercise relative to rest
			# slope
			the slope of the peak exercise ST segment
				# ca
				number of major vessels (0-3) colored by flourosopy
					# thal
					1 = normal; 2 = fixed defect; 3 = reversable defect
						# target
						1 or 0

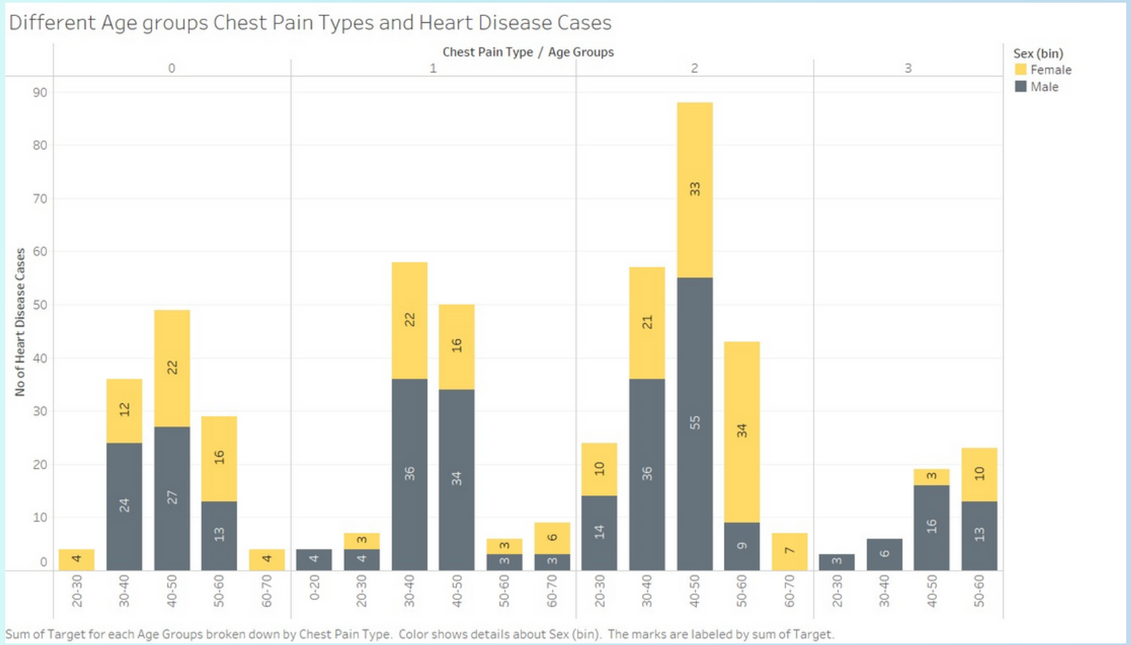
71 – 202	0 – 1	0 – 6.2	0 – 2	0 – 4	0 – 3	0 – 1
168	0	1	2	2	3	0
155	1	3.1	0	0	3	0
125	1	2.6	0	0	3	0
161	0	0	2	1	3	0
106	0	1.9	1	3	2	0
122	0	1	1	0	2	1
140	0	4.4	0	3	1	0
145	1	0.8	1	1	3	0
144	0	0.8	2	0	3	0
116	1	3.2	1	2	2	0
125	0	1.6	1	0	2	1

DATA VISUALISATION

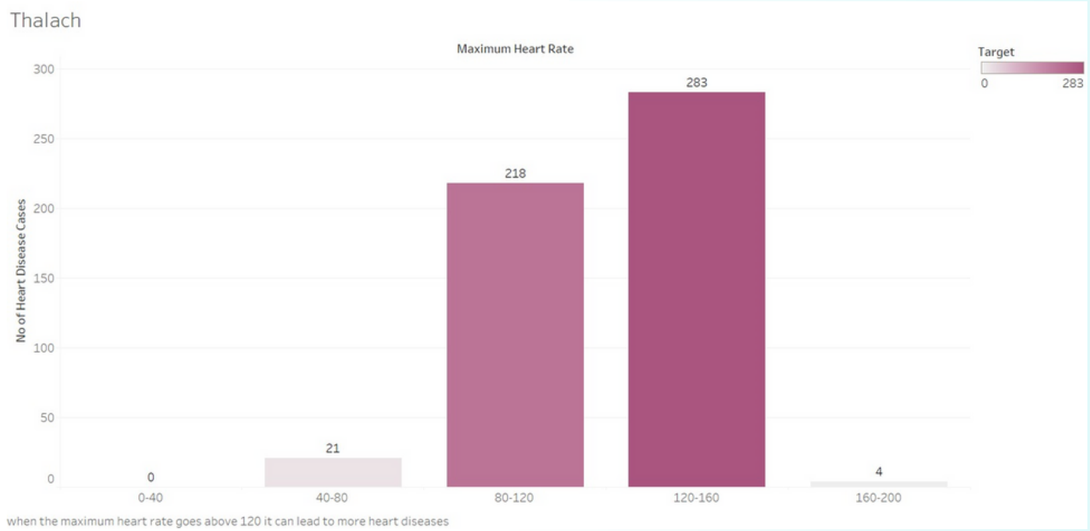


The above graph shows us the impact of heart disease across different age groups for both the genders (male and female).

The below graph shows us the number of heart disease cases across different age groups who have a symptom of **chest pain**.

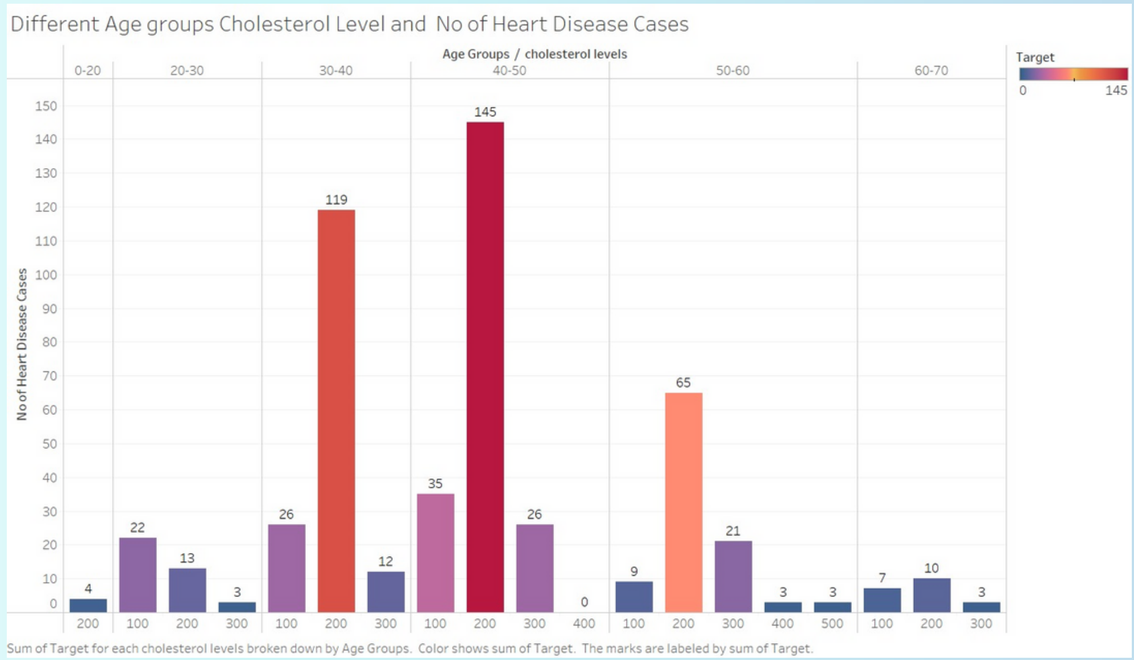


DATA VISUALISATION

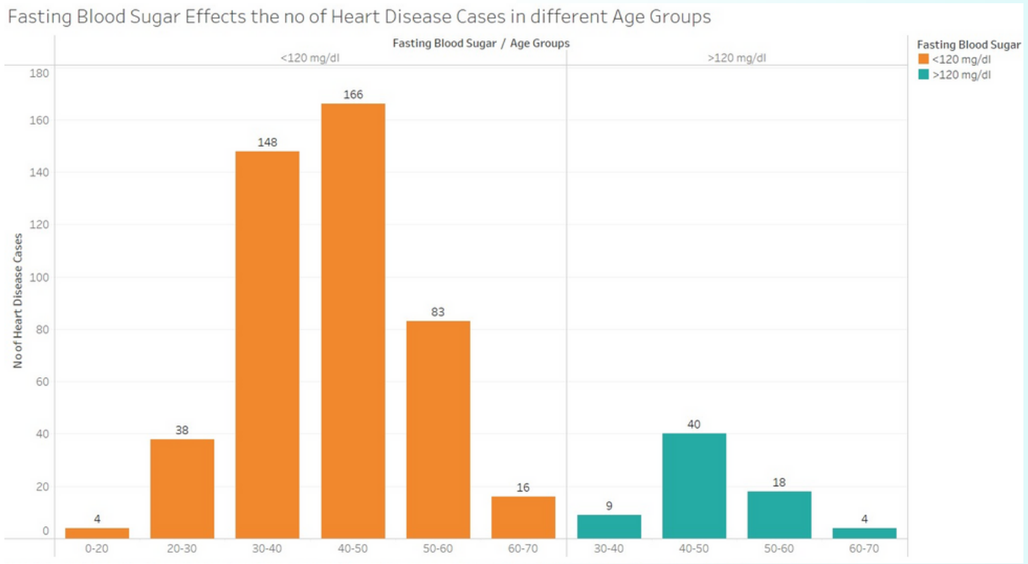


The above graph shows us the number of heart disease cases for different ranges of **maximum heart rate**.

The below graph shows us the number of heart disease cases across various age groups corresponding to their **cholesterol levels**.

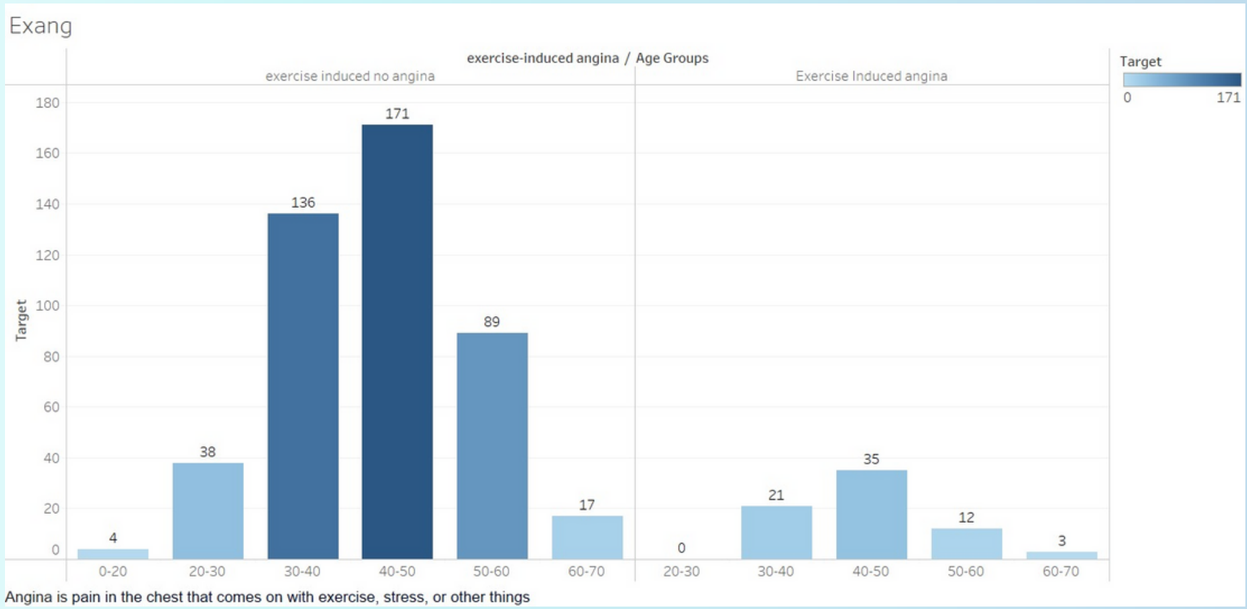


DATA VISUALISATION



The above graph shows us the number of heart disease cases for different **sugar levels in the blood after an overnight fasting** across the age groups.

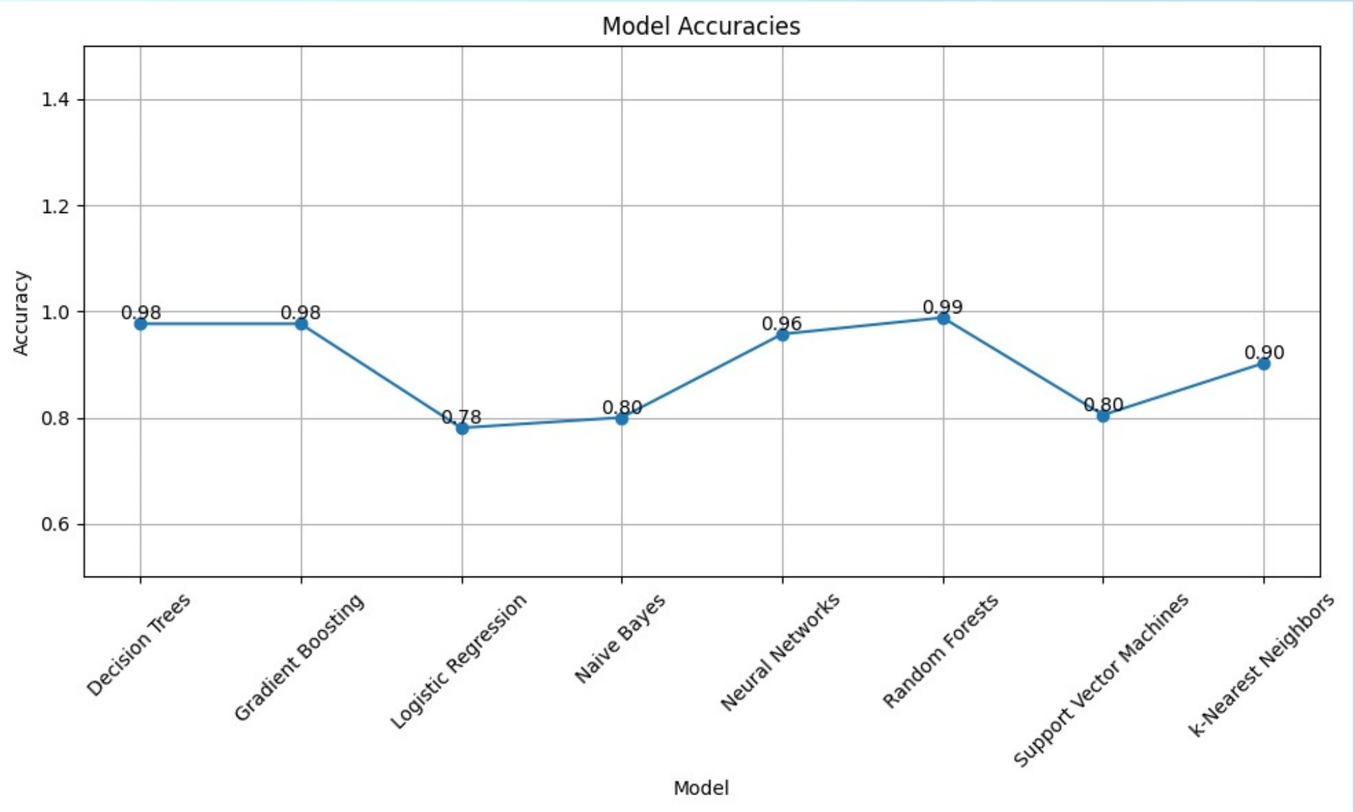
The below graph shows us the number of heart disease cases for different age groups, who **encountered angina while exercising**.



MODEL SELECTION

- We've selected ***Random Forest model*** for cardiovascular heart disease prediction.

- Among all the models on which we have trained and tested our data, we've finalised this model due to its ***exceptional performance (model accuracy), versatility, and scalability.***



MODEL SELECTION CODE -https://drive.google.com/file/d/1VgpRIhRogJDmauvshx6eQilYxn-19aL2/view?usp=drivesdk

RANDOM FOREST CLASSIFIER

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)

# Calculate accuracy
test_accuracy = accuracy_score(y_test, y_pred)

# Print the results
print("Test Accuracy:", test_accuracy)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Test Accuracy: 1.0
Confusion Matrix:
[[132   0]
 [  0 125]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       132
           1       1.00      1.00      1.00       125

    accuracy                           1.00       257
   macro avg       1.00      1.00      1.00       257
weighted avg       1.00      1.00      1.00       257
```

- Random Forest model is performing exceptionally well on this dataset, with high accuracy of 1, precision, recall, & F1-score.
- Since test accuracy is high and data is overfitting, we wanted to reduce this issue.

HYPER PARAMETER TUNING

```python
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_rf_model = grid_search.best_estimator_

# Fit the best model to the training data
best_rf_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = best_rf_model.predict(X_test)

# Calculate accuracy
test_accuracy = accuracy_score(y_test, y_pred)

# Print the results
print("Best Hyperparameters:", grid_search.best_params_)
print("Test Accuracy:", test_accuracy)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2,
Test Accuracy: 0.9883268482490273
Confusion Matrix:
[[132   0]
 [  3 122]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       132
           1       1.00      0.98      0.99       125

    accuracy                           0.99       257
   macro avg       0.99      0.99      0.99       257
weighted avg       0.99      0.99      0.99       257
```

- The hyperparameter tuning process led to apply the best hyperparameters for the Random Forest classifier
- The improvements from hyperparameter tuning can be observed in the model's performance metrics
- Before tuning, the model is overfitting the data due to suboptimal hyperparameters with test accuracy of 1
- However, after tuning, the model's accuracy is 0.98, precision, recall, and F1-score improved, leading to better overall performance

HANDLED THE OVERFITTING

To address overfitting in a Random Forest classifier, we applied few strategies, such as limiting tree depth, increasing minimum sample requirements, and using cross-validation for hyperparameter tuning from there we got a **Test accuracy of 0.9766536964980544** with an **Optimal number of trees: 95**

```
Optimal number of trees: 95
Test Accuracy: 0.9766536964980544
Confusion Matrix:
[[129   3]
 [  3 122]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       132
           1       0.98      0.98      0.98       125
```

```python
# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Limit the maximum depth of the trees to control overfitting
rf_classifier.set_params(max_depth=10)

# Increase minimum sample requirements to prevent overfitting
rf_classifier.set_params(min_samples_split=5, min_samples_leaf=2)

# Perform cross-validation to find the optimal number of trees
cv_scores = cross_val_score(rf_classifier, X_train, y_train, cv=5)

# Get the average cross-validation score and choose the number of trees accordingly
n_estimators = int(cv_scores.mean() * 100)

# Set the number of trees for the final model
rf_classifier.set_params(n_estimators=n_estimators)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)

# Calculate accuracy
test_accuracy = accuracy_score(y_test, y_pred)

# Print the results
print("Optimal number of trees:", n_estimators)
print("Test Accuracy:", test_accuracy)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

TRAINING MODEL WITH REGULARIZATION

- We have used ***Regularization techniques to prevent overfitting in Random forest.*** It adds a penalty term to the model's cost function to discourage it from fitting the noise in the training data and to promote a simpler and more generalizable model.
- The results also suggests that **L1 regularization** is more effective in reducing overfitting and achieving better accuracy than **L2 regularization.**

```
Accuracy with L1 Regularization (Random Forest): 0.9688715953307393
Accuracy with L2 Regularization (Random Forest): 0.914396887159533
Confusion Matrix with L1 Regularization (Random Forest):
[[127   5]
 [  3 122]]
Confusion Matrix with L2 Regularization (Random Forest):
[[119  13]
 [  9 116]]
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Step 2: Data Preprocessing (if needed)
# If needed, perform data preprocessing steps here, such as encoding categorical variables, scaling, etc.

# Step 3: Train-Test Split
X = df.drop(columns=['target'])
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Step 4: Train the Random Forest Classifier with Regularization
# Apply regularization hyperparameters
model_rf_l1 = RandomForestClassifier(min_samples_split=5, min_samples_leaf=2, max_features='sqrt', random_state=42)
model_rf_l1.fit(X_train, y_train)

model_rf_l2 = RandomForestClassifier(min_samples_split=10, min_samples_leaf=5, max_features='log2', random_state=42)
model_rf_l2.fit(X_train, y_train)

# Step 5: Evaluate the models
y_pred_rf_l1 = model_rf_l1.predict(X_test)
y_pred_rf_l2 = model_rf_l2.predict(X_test)

accuracy_rf_l1 = accuracy_score(y_test, y_pred_rf_l1)
accuracy_rf_l2 = accuracy_score(y_test, y_pred_rf_l2)

print("Accuracy with L1 Regularization (Random Forest):", accuracy_rf_l1)
print("Accuracy with L2 Regularization (Random Forest):", accuracy_rf_l2)

# Step 6: Calculate and compare the confusion matrices
conf_matrix_rf_l1 = confusion_matrix(y_test, y_pred_rf_l1)
conf_matrix_rf_l2 = confusion_matrix(y_test, y_pred_rf_l2)

print("Confusion Matrix with L1 Regularization (Random Forest):")
print(conf_matrix_rf_l1)

print("Confusion Matrix with L2 Regularization (Random Forest):")
print(conf_matrix_rf_l2)
```

DATA AUGMENTATION

- **To increase the size and diversity of the training dataset** by creating additional samples from the existing data and minimizing overfitting, especially when the original dataset is small or imbalanced we need to introduce variations in the data, we used data augmentation,which allows the model to generalize better to unseen data and learn more robust features.
- The confusion matrix shows that the model correctly classified 132 true negatives and 115 true positives. It made 10 false predictions (false negatives and false positives). Overall, the model is doing well in classifying both positive and negative cases, but it has a slightly higher number of false negatives (10 cases where individuals with heart disease were misclassified as not having heart disease)

```
Test Accuracy: 0.9610894941634242
Confusion Matrix:
[[132   0]
 [ 10 115]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       132
           1       1.00      0.92      0.96       125
```

```python
# Concatenate the augmented data with the original data
X_train_augmented = pd.concat([X_train[y_train == 0], X_train_augmented])
y_train_augmented = pd.concat([y_train[y_train == 0], y_train_augmented])

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Perform GridSearchCV to find the best hyperparameters
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train_augmented, y_train_augmented)

# Get the best hyperparameters
best_rf_model = grid_search.best_estimator_

# Fit the best model to the training data
best_rf_model.fit(X_train_augmented, y_train_augmented)

# Make predictions on the test data
y_pred = best_rf_model.predict(X_test)

# Calculate accuracy
test_accuracy = accuracy_score(y_test, y_pred)
```

RESULTS & CONCLUSIONS

- We conducted a predictive analysis using a **Random Forest classifier to predict the presence of heart disease** based on various cardiovascular risk factors.

- **To prevent overfitting, we employed L1 and L2 regularization techniques**. The model achieved an accuracy of 96.11% on the test set and demonstrated high precision, recall, and F1-score for both classes.

- We also **performed hyperparameter tuning using cross-validation** and obtained the best hyperparameters (max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50).

- **Data augmentation was utilized to increase the dataset's diversity** and enhance the model's generalization ability.

- The Random Forest classifier showed robust performance in correctly classifying individuals with and without heart disease, making it a reliable tool for this classification task.

PROJECT CODE-https://drive.google.com/file/d/1VfzwoS0EBFWsTMn7ss8zNeOnDPnhcPzl/view?usp=drivesdk

THANK YOU

TEAM NEON

IBM SkillsBuild

CSRBOX