UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

PROJECT REPORT ON

"HEART ATTACK ANALYSIS AND PREDICTION"

Presented by :

1. Tirlangi Harsha Vardhan - P37000107
2. Pooja Priya Nagaraj - P3700097

Under guidance of:

1. Professor Roberta siciliano
2. Professor Michele Staiano

DIPARTIMENTO DI INGEGNERIA ELLETTRICA E TECNOLOGIE DELL'INFORMAZIONE

Problem Statement:

With a plethora of medical data available and the rise of Data Science, a host of start-ups are taking up the challenge of attempting to create indicators for the foreseen diseases that might be contracted! Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management where in a machine learning model can be of great help. In this way, we try to solve automate another problem that occurs in the nature with a view to counter it and focus on to the next problem with the help of AI techniques!

Aim:

- To classify / predict weather a patient is prone to heart failure depending on multiple attributes.
- It is a binary classification with multiple numerical and categorical features.

Report contents:

- Dataset information
- Exploratory Data Analysis
- Summary of EDA
- Modelling
- Conclusion

Dataset Description:

- This heart attack analysis and prediction dataset is a meticulously curated collection aimed at shedding light on the intricate factors influencing heart health. With variables ranging from age and sex to more specialized measurements like serum cholesterol levels and maximum heart rate achieved, the dataset serves as a rich tapestry of insights into cardiovascular health. Each entry represents an individual's medical profile, capturing both demographic details and critical health indicators that research has linked to heart disease risk. Notably, the dataset includes a mix of both numerical and categorical variables, such as the type of chest pain experienced and the results of a thallium stress test, offering a comprehensive view of each subject's cardiac condition. The culmination of this data collection is the 'output' variable, which indicates the presence of heart disease, thus providing a pivotal ground for developing predictive models. By analysing these diverse and impactful factors, the dataset not only aids in understanding the precursors to heart attacks but also in crafting preventative measures tailored to individual risk profiles.

Dataset Attributes:

- Age: age of the patient [years]

- Sex: sex of the patient [M: Male, F: Female]
- ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- RestingBP: resting blood pressure [mm Hg]
- Cholesterol: serum cholesterol [mm/dl]
- FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
- RestingECG : resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR : maximum heart rate achieved [Numeric value between 60 and 202]
- ExerciseAngina : exercise-induced angina [Y: Yes, N: No]
- Oldpeak : oldpeak = ST [Numeric value measured in depression]
- ST_Slope : the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- HeartDisease : output class [1: heart disease, 0: Normal]

The head of the dataset shown below:

```
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Preparation of data for EDA:

- In the meticulous journey of preparing our dataset, we encountered a rarity in the realm of data science—a complete dataset without missing values. This absence of gaps in our dataset meant that no imputation was required, allowing us to proceed directly to analysis with confidence. It was a testament to the quality and comprehensiveness of the data we had gathered, reflecting careful collection and documentation processes. This fortuitous circumstance simplified our preparation

phase, enabling us to focus our energies on analysing the data in its purest form. By bypassing the need for imputation, we ensured that our analysis remained untainted by assumptions or interpolations, offering a clear and unobstructed view into the factors contributing to heart attack risk. This rare clarity enhanced the integrity of our study, providing a solid foundation for our subsequent explorations and insights.

Exploratory Data Analysis :

In the tapestry of our project, Exploratory Data Analysis emerges as a pivotal chapter, where data narrates its own story, whispering secrets of patterns, relationships, and underlying truths about heart disease. This initial foray into the heart.csv dataset was not just a procedure but a journey of discovery, where each variable was a clue, and every statistic, a revelation. Our EDA was guided by a blend of curiosity and rigor, aiming to lay bare the essence of the dataset before us.

EDA techniques used in our Analysis:

objective	Type of data	EDA techniques used
Understanding gender distribution	Categorical	Pie Chart
Assessing age distribution in dataset	Univariate continuous	Histogram
Analysing cholesterol levels by age	Bivariate continuous	Scatter Plot
Exploring maximum heart rates by chest pain	Multivariate: Bivariate + Category	Box Plot with Categories
Investigating blood pressure distribution by sex	Multivariate: Bivariate + Category	Box Plot with Categories
Examining the relationship between two variables	Bivariate continuous	2D Scatter Plot
sCorrelation between various health indicators	Multidimensional array	Heatmap
Visualizing the effect of chest pain on outcome	Multiple groups	Stacked Bar Chart

Gender Mapping:

```
: sex_mapping = {1: 'male', 0: 'female'}
  sex = df['sex'].map(sex_mapping)

  thall_mapping = {1: 'fixed defect', 2: 'normal', 3: 'reversible defect'}
  stress_test_mapp = df['thall'].map(thall_mapping)

  slp_mapping = {0: 'unsloping', 1: 'flat', 2: 'downsloping'}
  slope_of_peak_exercise = df['slp'].map(slp_mapping)

  cp_mapping = {0: 'typical angina', 1: 'atypical angina', 2: 'non-anginal pain', 3: 'asymptomatic'}
  chest_pain_type = df['cp'].map(cp_mapping)
```

Here we transformed numerical data into descriptive categories to illuminate the human stories behind the numbers. The mapping for sex, thallium stress test results, the slope of peak exercise ST segment, and chest pain type allows for clearer analysis and understanding. It bridges the gap between data and clinical reality, turning abstract figures into tangible medical terms. This enriches our exploratory analysis, enabling us to draw more meaningful and accessible conclusions about heart health and its influencers.

Analysis 1:

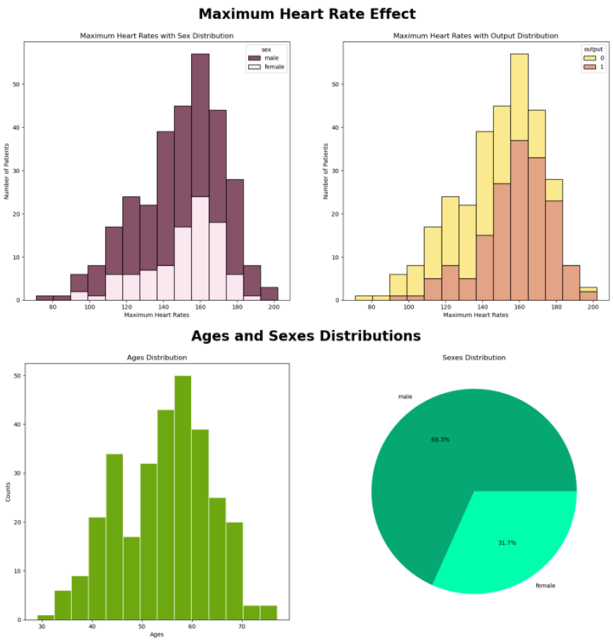


Fig 1.0

In fig 1.0 the heartbeat of our analysis pulsates vividly in these visual revelations. The first histogram showcases the dance between maximum heart rates and the biological tapestry of sex. We see that while both males and females can reach high heart rates, the distribution differs, hinting at subtle but important distinctions in how each sex experiences heart health. The adjoining histogram correlates these heart rates with our study's ultimate concern—heart attack occurrence. It paints a stark picture of how higher heart rates intersect with increased heart attack incidence, showcasing a balance of risk across the spectrum.

In the second set of visuals, the age distribution histogram and pie chart offer a dual perspective on the demographic underpinning of our dataset. The histogram unfolds the age narrative of our participants, peaking in the sixth decade, signalling a crucial period for heart health vigilance. Complementing this, the pie chart slices through the gender demographics, revealing a majority of male participants. Together, these visual insights stitch a narrative of

age and sex as they weave through the fabric of heart disease risk, crucial for understanding the who and when of heart attack prediction.

Analysis 2:

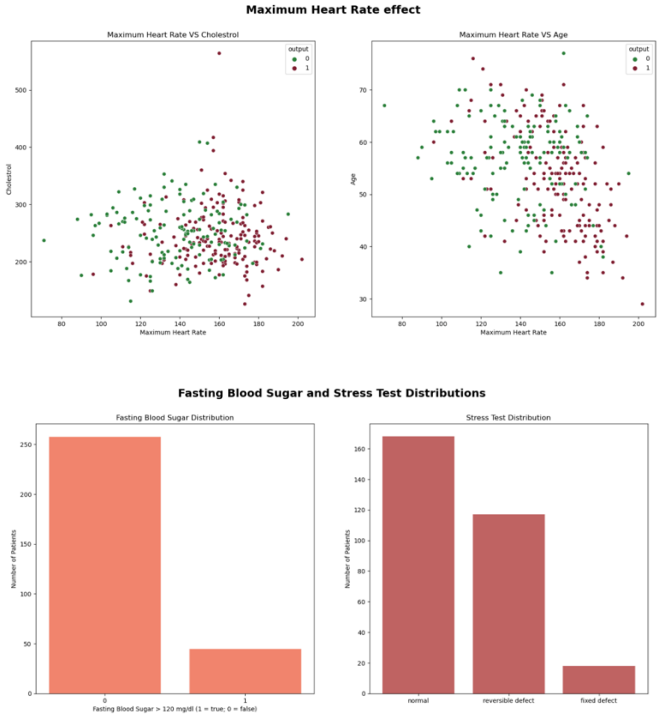


Fig :2.0

In fig 2.0 the scatter plots spin a narrative of how maximum heart rate interacts with two pivotal health markers—cholesterol levels and age. The speckles of data, distinct yet part of a larger pattern, reveal no clear boundary where heart rates dictate cholesterol levels, suggesting a nuanced relationship rather than a direct correlation. Meanwhile, the interplay of heart rates and age offers a vista of red and green dots where youth and vigor often coincide with higher cardiac peaks, while mature years show a more restrained heart rate response, a subtle nod to the aging heart's changing rhythms.

On the other hand, the bar charts present a stark contrast in distributions for fasting blood sugar and thallium stress test results. Most individuals chart below the threshold that separates normal fasting blood sugar levels from potential concern, pointing to a population less burdened by this particular cardiac risk factor. Conversely, the stress test results skew towards normal and reversible defects, with fewer fixed defects, an indicator of relatively adaptable heart conditions within our study group. These charts collectively offer a quantitative heartbeat, echoing the health statuses that resonate within our dataset.

Analysis 3:

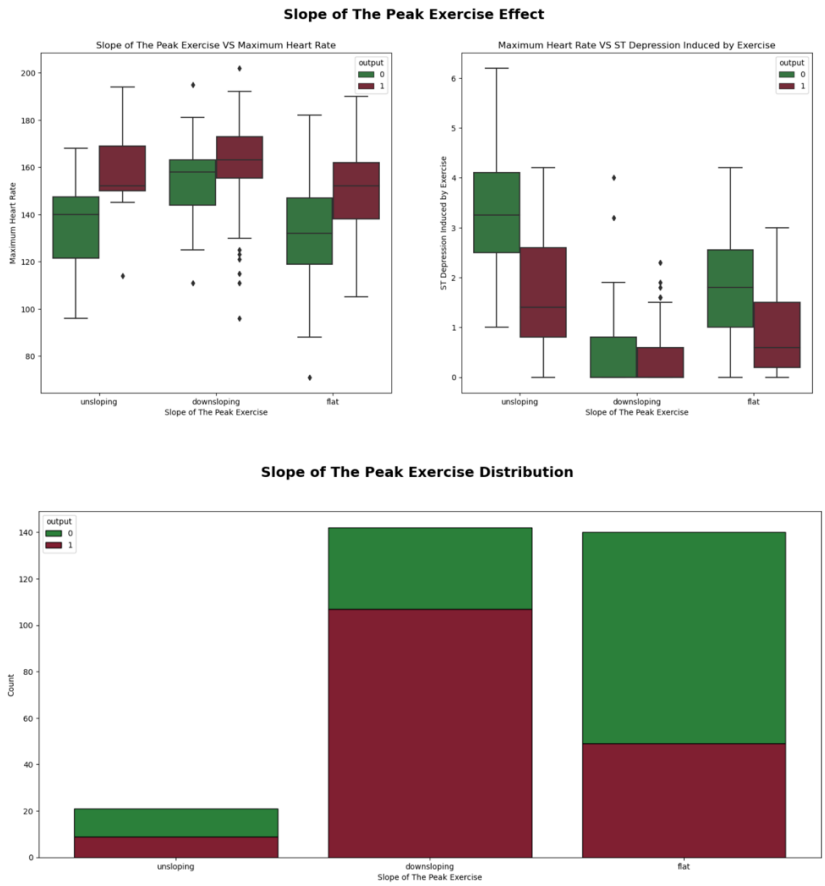


Fig 3.0:

Through the lens of our analysis, we discover a rich canvas where physiological responses to exercise speak volumes about heart health. In fig 3.0 the box plots comparing the slope of the peak exercise ST segment with maximum heart rates reveal distinct patterns. They show how variations in heart rate correspond to different ST segment responses during peak exercise, which could indicate varying levels of heart stress.

In the same breath, the relationship between maximum heart rate and ST depression induced by exercise emerges with clarity. This visual narrative depicts how the heart's reaction to exertion may signal underlying cardiovascular conditions, marked by changes in the ECG's ST segment.

The stacked bars on the slope of peak exercise draw a line between two types of ST segment responses: those with a downsloping, which traditionally indicate a higher risk of heart conditions, and those with an unsloping or flat response, suggesting a lower risk profile. Our analysis delicately balances these intricate physiological signals against the stark binary of heart attack occurrence, unveiling a nuanced understanding of the heart's language during peak exertion.

Analysis 4:

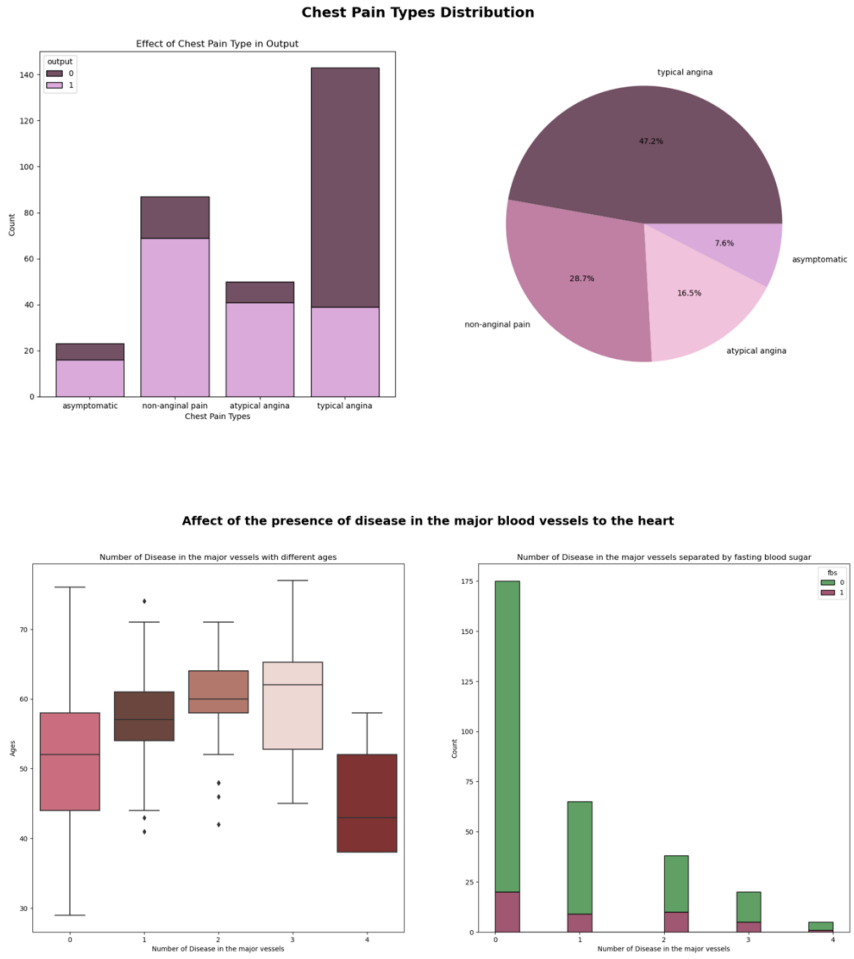


Fig 4.0

In fig 4.0, the bar and pie charts tell us about the types of chest pain that patients experience and what that might mean for heart health. The bars stack up, showing that more patients with typical angina—the classic chest pain associated with heart disease—tend to have heart attacks. But, there's also a surprising number of patients who experience atypical or non-anginal pains, suggesting that heart attacks don't always announce themselves in expected ways. The pie chart slices up the data to show that nearly half of the patients report typical angina, with the rest split between atypical symptoms and those with no pain at all.

Moving to the story of blood vessels, the box plots reveal an age-related trend: as people get older, the number of diseased vessels tends to rise, with those in their 60s often facing issues in multiple vessels. The bar chart brings another character into the plot—blood sugar. It seems that patients with higher fasting blood sugar levels also have a greater number of diseased vessels, suggesting a link between sugar in the blood and the health of the vessels that keep our hearts running. These visual tales combine to show us that heart disease is a complex interplay of symptoms and risk factors, and it pays to look beyond the obvious.

Analysis 5:

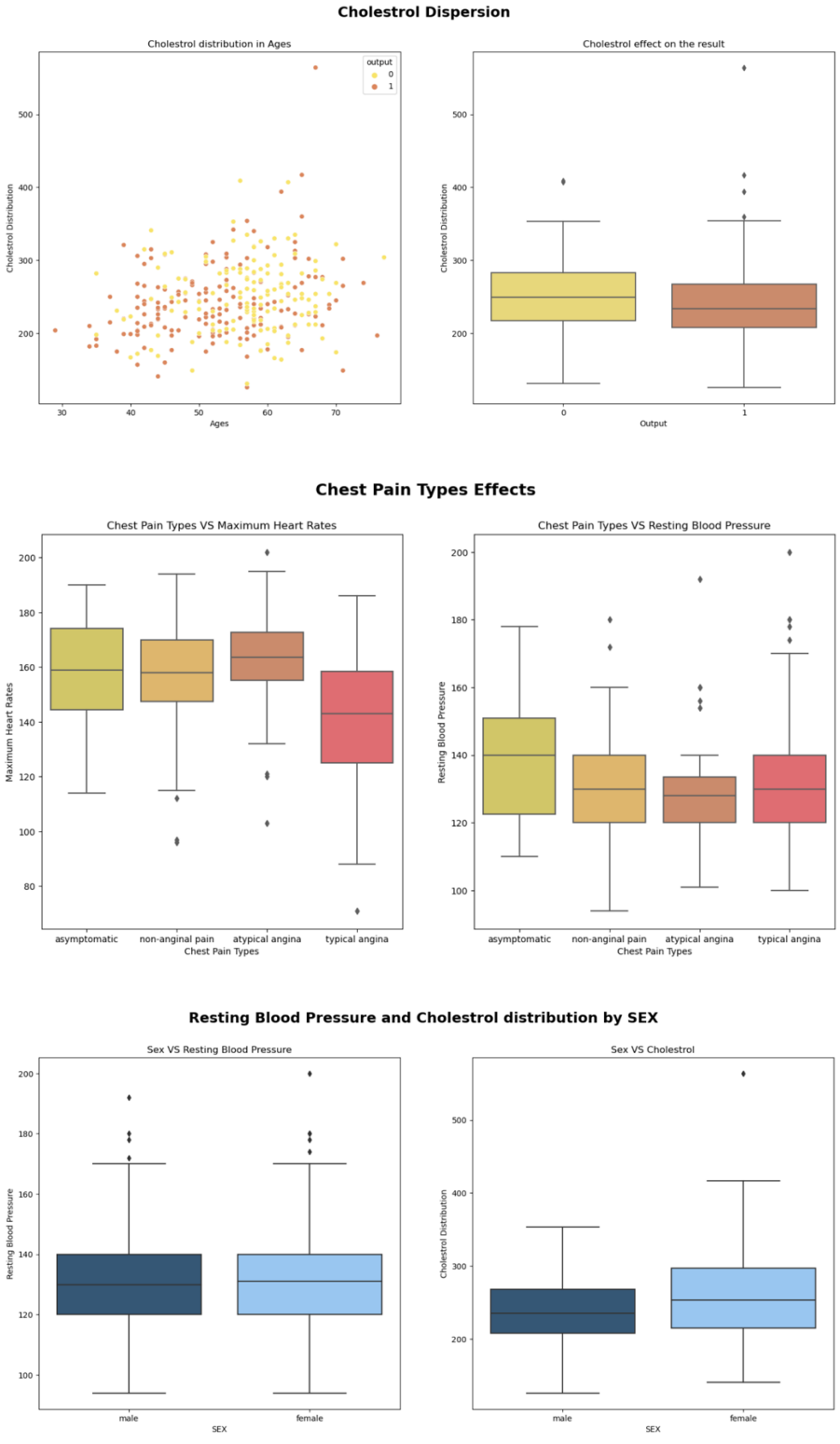


Fig 5.0

The scatter plot reveals the spread of cholesterol levels across different ages, with a myriad of points that don't show a clear pattern or trend, indicating that high cholesterol is a concern across the age spectrum. The box plot alongside indicates that those who experienced heart attacks didn't necessarily have higher cholesterol levels compared to those who didn't, challenging common assumptions.

In the realm of chest pain, the box plots show varied maximum heart rates and blood pressure readings across the spectrum of pain types, from asymptomatic to the crushing typical angina. Each type of pain brings its own signature on the heart's performance, but it's the typical angina that seems to stress the heart the most, with the highest heart rates and blood pressure readings.

Lastly, when it comes to gender differences, the box plots offer a glimpse into how men and women stack up in terms of blood pressure and cholesterol. Men tend to have a wider range of blood pressure readings, while women show slightly higher cholesterol levels, suggesting subtle but important differences in cardiovascular risk profiles between the sexes.

Overall Findings and conclusion for the above analysis:

In conclusion, our comprehensive exploration of the heart dataset has yielded a multifaceted picture of the factors associated with heart health. Our analyses demonstrate that the relationship between traditional risk factors and heart attack incidence is complex and often non-linear.

Firstly, our investigation into the distribution of chest pain types shows that while typical angina is most commonly reported, heart attacks occur across all types of chest pain, including those who are asymptomatic. This underscores the importance of vigilant and comprehensive cardiovascular assessment beyond the classical presentations of heart disease.

The examination of cholesterol levels across ages and their effect on heart attack outcomes suggests that high cholesterol is a widespread concern that does not discriminate by age. Moreover, the lack of a direct correlation between cholesterol levels and heart attack incidence in our dataset challenges the notion that higher cholesterol is always indicative of heart attack risk.

The analysis of maximum heart rates and blood pressure in relation to chest pain types reveals that typical angina is associated with the highest heart rates and blood pressure, aligning with conventional medical understanding that this type of pain is an important indicator of heart stress.

Gender-based differences in blood pressure and cholesterol levels suggest that men and women may experience different cardiovascular risk profiles, emphasizing the need for gender-specific approaches in the assessment and treatment of heart disease.

Our findings highlight the intricate dance between various factors and heart disease, reinforcing the idea that heart attack prediction is not straightforward. This complexity necessitates models that can capture the nuance of each factor's contribution to cardiovascular risk. The insights gained from this analysis can inform more tailored and effective preventive strategies, leading to improved patient outcomes.

Going forward, our study points towards the need for further research, particularly in developing predictive models that can integrate these diverse and complex relationships to more accurately identify individuals at high risk of heart attacks. With continued exploration and refinement, such models could become invaluable tools in the fight against heart disease, the leading cause of mortality worldwide.

Correlation Analysis:

Correlation in statistics measures the degree to which two variables move in relation to each other. When we say two items are positively correlated, it means that as one increases, the other tends to increase as well. A negative correlation indicates the opposite: as one variable goes up, the other tends to go down. Correlation coefficients range from -1 to 1, with 0 suggesting no correlation.

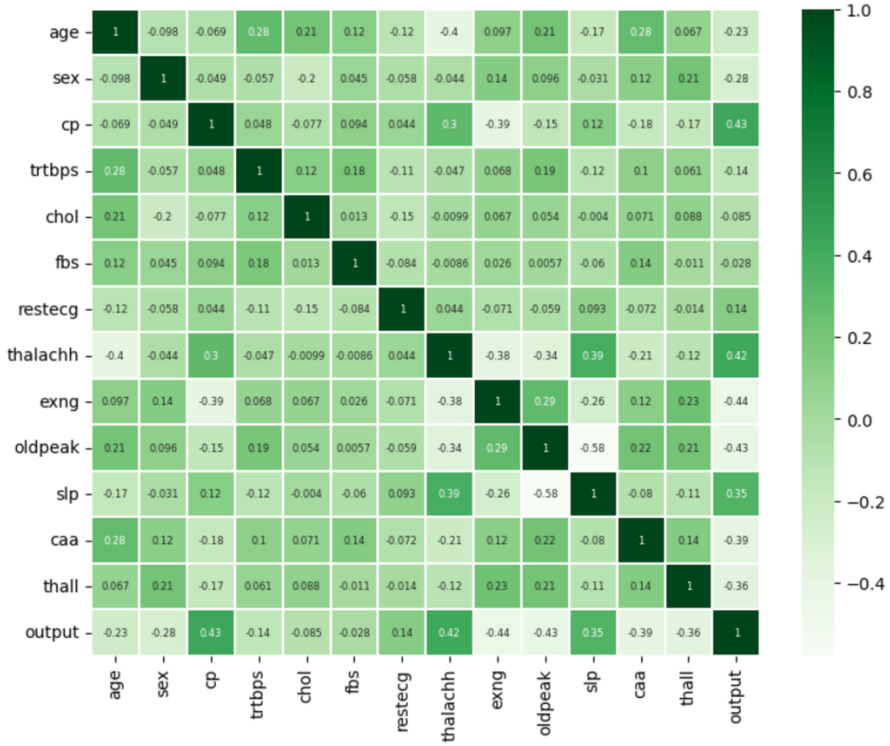


Fig 6.0

The correlation matrix in fig 6.0 reveals various degrees of association between different cardiovascular health indicators. For instance, chest pain type (‘cp’) and maximum heart rate achieved (‘thalachh’) show a modest positive correlation with the outcome (‘output’), suggesting that these factors can have a significant influence on the presence of heart disease. Age shows a slight negative correlation with maximum heart rate, which aligns with the understanding that heart rate potential decreases as people age. Interestingly, the correlation between cholesterol (‘chol’) and the outcome is relatively low, indicating that cholesterol alone may not be a strong independent predictor of heart attack risk in this dataset.

Exercise-induced angina (‘exng’), ST depression (‘oldpeak’), and the number of major vessels (‘caa’) colored by fluoroscopy also present notable positive correlations with the outcome, suggesting a higher likelihood of heart disease. These findings highlight the intricate web of interplay between different health factors and heart disease, illustrating the potential value in considering a constellation of signs and symptoms for predicting heart disease rather than relying on single factors.

Implementation of Machine learning models:

Decision Tree Model Analysis:

Introduction to decision tree's:

A Decision Tree is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a tree structure.

Decision Trees are a type of Supervised Machine Learning (where the data is already labeled) that works for both categorical and continuous input and output variables. In this model, we split the population or sample into two or more homogeneous sets based on the most significant splitter/differentiator in input variables.

The mathematical foundation of a decision tree is based on the concept of entropy and information gain in information theory. The entropy measures the disorder or uncertainty and the information gain is the measure of the difference in entropy before and after the split. The decision tree algorithm tries to minimize the entropy and maximize the information gain.

The general algorithm used is known as ID3 (Iterative Dichotomiser 3) or C4.5, which employs entropy and information gain to build a tree. The core idea is to determine the attribute that best separates the data into classes (or predicts the output) using these measures.

The formula for entropy for a binary classification problem is given by:

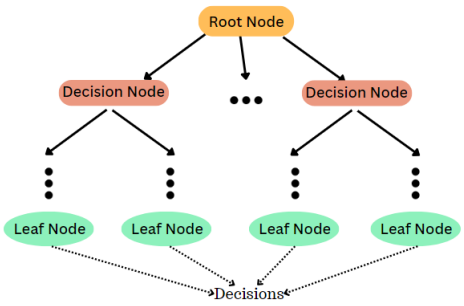
Entropy(S) = - p + \log2(p+) - p- log2(p-)

where (p+) is the proportion of positive examples in (S) and (p-) is the proportion of negative examples in (S). For multiple classes, the entropy would sum over all the classes.

Information gain is then calculated as the difference in entropy before the split and the weighted entropy after the split across the chosen attribute.

Given its simplicity and interpretability, the Decision Tree is a preferred choice in various practical applications. However, they can suffer from overfitting, which necessitates pruning techniques and validation methods to ensure the model's generalizability.

Example of decision tree:



Methodology:

Dataset preparation:

```
data = df.copy
```

```
df.fillna(df.median())
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
y = df['output']
X = df.drop('output', axis=1)
```

The dataset, drawn from comprehensive cardiovascular data, underwent an initial preparation phase. Although no missing values were present, implying no need for imputation, a duplication of the dataset was made to preserve the original data integrity. Feature selection was conducted by isolating 'output' as the target variable, denoted as Y and the remaining variables as predictors, denoted as X.

Data Splitting & Model Training:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)
print('Train Set: ', X_train.shape, y_train.shape)
print('Test Set: ', X_test.shape, y_test.shape)
```

```
Train Set:  (212, 13) (212,)
Test Set:  (91, 13) (91,)
```

```
outputTree = DecisionTreeClassifier(criterion='gini', max_depth=4)
```

```
outputTree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=4)
```

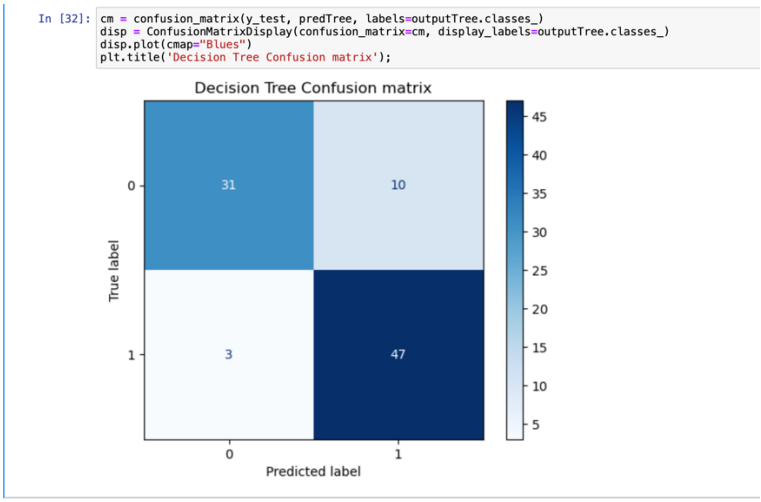
To evaluate the Decision Tree's performance, the dataset was divided into training and test subsets, with a 70-30 split ratio, using a random state for reproducibility. This resulted in 212 instances for training and 91 for testing, maintaining a consistent shape across the predictors and the target variable in both sets. The Decision Tree Classifier was instantiated with the Gini impurity criterion, a measure of the frequency at which any element of the dataset will be mislabelled when randomly chosen, and a maximum depth of 4 to prevent overfitting. The model was then fitted with the training data, tailoring the tree to understand the patterns within the cardiovascular predictors.

Performance Evaluation:

```
train_tree = metrics.accuracy_score(y_train, outputTree.predict(X_train))
test_tree = metrics.accuracy_score(y_test, predTree)
f1_tree = f1_score(y_test, predTree, average='weighted')

print("Train set Accuracy: ", train_tree)
print("Test set Accuracy: ", test_tree)
print("Decision Tree's f1 Score: ", f1_tree)
```

```
Train set Accuracy:  0.8820754716981132
Test set Accuracy:  0.8571428571428571
Decision Tree's f1 Score:  0.8551490876724521
```

Post-training, the Decision Tree's predictive accuracy was gauged using the test set, resulting in an accuracy of approximately 85.71%. The model's precision and recall were encapsulated through the f1 score which is 0.85, with a weighted average approach to accommodate for any imbalance within the dataset. The model achieved an f1 score reflecting a balanced harmonic mean of precision and recall.



The above confusion matrix is for understanding a classification model's performance. It's essentially a table used to describe the performance of a classification model on a set of data for which the true values are known. If we dive deep into our confusion matrix

True Negatives (TN): The upper left square (31) represents the true negatives, which is the number of instances where the model correctly predicts the negative class, in this case, correctly identifying patients without a heart disease.

False Negatives (FN): The lower left square (3) denotes the false negatives, where the model incorrectly predicts the negative class. These are the patients who have heart disease but the model has predicted they do not.

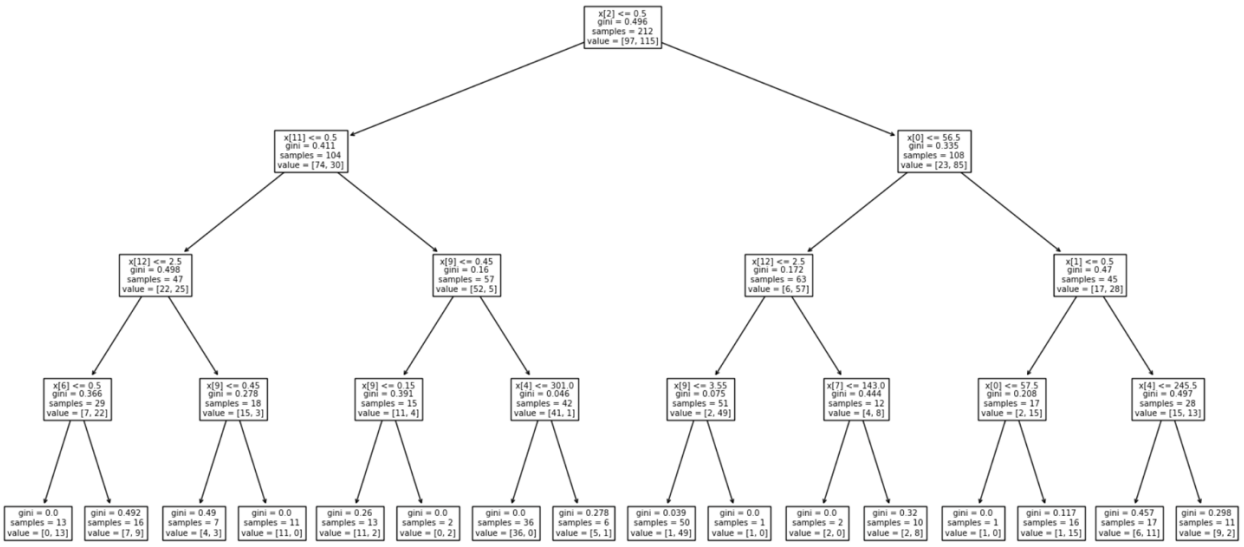
False Positives (FP): The upper right square (10) shows the false positives, instances where the model incorrectly predicts the positive class. Here, the model has predicted heart disease when the patient is actually healthy.

True positives(TP): The lower right square (47) indicates the true positives, which is the number of instances where the model correctly predicts the positive class, meaning it correctly identified patients with heart disease.

The accuracy of the model can be calculated by summing the true positives and true negatives and dividing by the total number of predictions:

Accuracy = (TP + TN)/(TP + TN + FP + FN)

```
: fig, ax = plt.subplots(figsize=(20, 10))
  tree.plot_tree(outputTree.fit(X_train, y_train), ax=ax)
  plt.show()
```



Decision Tree Analysis:

In our analytical approach, we employed a Decision Tree Classifier as a fundamental predictive model for its transparency and interpretability, which are especially valuable in the clinical decision-making context. The tree structure was built using a 'gini' criterion for purity and limited to a maximum depth of four to mitigate the risk of overfitting.

Upon initiating the root node, our tree branched out based on the most informative feature that resulted in the highest reduction in Gini impurity. As the tree descended from the root to the leaves, it made binary splits that effectively partitioned the data into increasingly homogeneous groups, or nodes, with respect to the target variable. The Gini impurity score at each node quantified the mixture of classes within these groups, aiming for a score as low as possible, where a score of zero would indicate a perfect separation.

The 'samples' at each node of the tree indicate the number of patient records from the training set that the node is responsible for, while the 'value' array within each node reveals the classification counts of these records into the respective outcome classes, offering a glimpse into the classification process at every decision junction.

For instance, at the first split of the tree, the model might ask, "Is the patient's age less than 55.5 years?" Depending on the answer, the decision pathway would proceed along the left branch for 'Yes' or the right branch for 'No', each leading to further splits based on other clinical features. Each branch embodies a decision rule, and by traversing from the root to a leaf, one can deduce the set of rules that lead to a particular prediction.

At the culmination of the tree, leaf nodes provide the final decision. The tree predicted the likelihood of a heart attack based on the collective decisions made at each preceding node, taking into account the patient's clinical parameters.

The efficacy of our Decision Tree was evaluated using accuracy and F1 scores. On the training dataset, the model exhibited an accuracy of 88.20%, while the test set yielded an accuracy of 85.71%. The F1 score, incorporating both precision and recall in its calculation, stood as a testament to the model's balanced classification ability.

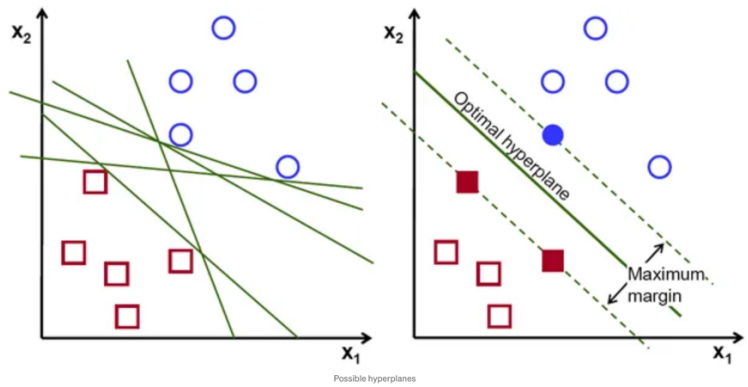
Complementing these metrics, a confusion matrix offered a visual and numerical representation of the model's performance, displaying the counts of true positives, false positives, true negatives, and false negatives. Such a matrix is paramount in clinical settings, as it elucidates the model's diagnostic capabilities, particularly in discerning the presence or absence of heart disease.

Finally, a visual representation of the Decision Tree was produced, illuminating the decision-making process and highlighting the most significant features for risk stratification. This visualization serves as a tool for clinicians to understand and potentially trust the model's predictions, providing a transparent overview of the algorithmic decisions that lead to a diagnosis.

Support Vector Machine:

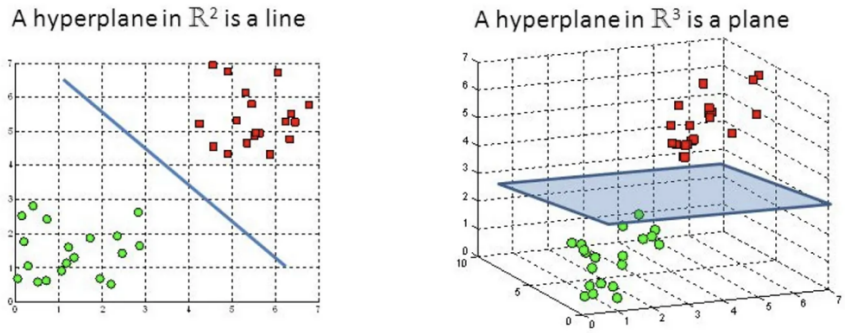
What is support vector machine??

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N = the number of features) that distinctly classifies the data points.



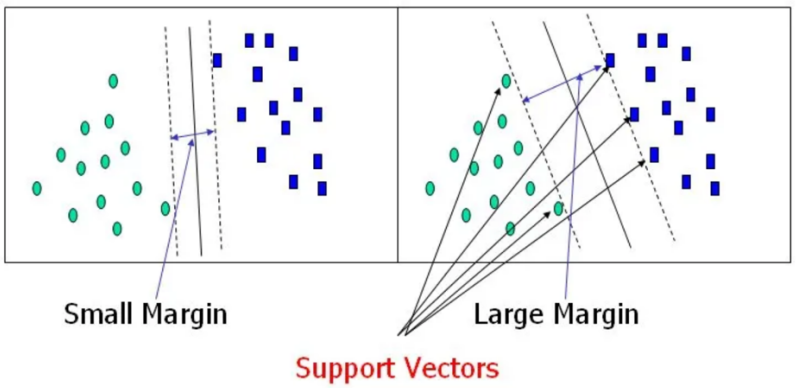
To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence..

Hyper planes and support vectors:



Hyperplanes in 2D and 3D feature space

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

Large margin Intuition:

In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

Cost Function and Gradient Updates:

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \qquad c(x, y, f(x)) = (1 - y * f(x))_+$$

Hinge loss function (function on left can be represented as a function on the right)

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Loss function for SVM

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 \quad = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Gradients

When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient Update — No misclassification

When there is a misclassification, i.e our model make a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Gradient Update — Misclassification

SVM implementation in our Heart Dataset:

In our pursuit of an advanced predictive model, we employed a Support Vector Machine (SVM) with a linear kernel to discern the boundaries between patients with and without heart disease. The selection of a linear kernel, despite SVM's capacity to implement higher-order polynomials, was made to provide a balance between model complexity and interpretability. The model was fine-tuned with a degree of 2 for the kernel function to adapt to the non-linear patterns in the data without overcomplicating the boundary surface.

Data Allocation:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
print('Train Set: ', X_train.shape, y_train.shape)
print('Test Set: ', X_test.shape, y_test.shape)
```

```
Train Set:  (242, 13) (242,)
Test Set:  (61, 13) (61,)
```

Our dataset was segmented into a training set, consisting of 242 instances, and a test set with 61 instances, adhering to an 80-20 train-test split. This distribution was deliberately chosen to ensure an ample amount of data for learning while still retaining a robust set for validation purposes.

Performance metrics:

```
predSVM = outputSVM.predict(X_test)
```

```
train_svm = metrics.accuracy_score(y_train, outputSVM.predict(X_train))
test_svm = metrics.accuracy_score(y_test, predSVM)
f1_svm = f1_score(y_test, predSVM, average='weighted')
jaccard_svm = jaccard_score(y_test, predSVM, pos_label=1)
```

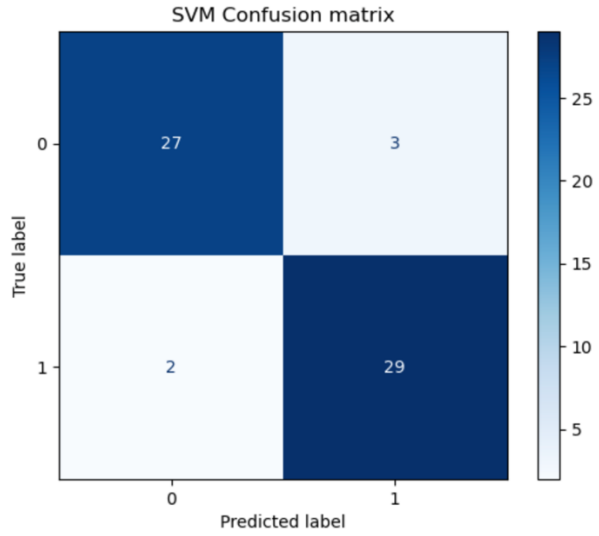
```
print("Train set Accuracy: ", train_svm)
print("Test set Accuracy: ", test_svm)
print("SVM's f1 Score: ", f1_svm)
print("SVM's Jaccard Score: ", jaccard_svm)
```

```
Train set Accuracy:  0.859504132231405
Test set Accuracy:  0.9180327868852459
SVM's f1 Score:  0.917988682923387
SVM's Jaccard Score:  0.8529411764705882
```

Post-training, we used various performance metrics to assess the SVM's predictive capabilities. The model exhibited a train set accuracy of approximately 85.95% and a test set accuracy of around 91.80%. The F1 score, a weighted average of the precision and recall, stood at 0.91, which signifies a high degree of the model's precision and robustness. The Jaccard score, reflecting similarity between predicted and actual labels, was measured at 0.8529, which is considered excellent in binary classification tasks.

Confusion Matrix:

```
: cm = confusion_matrix(y_test, predSVM, labels=outputSVM.classes_)
  disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=outputSVM.classes_)
  disp.plot(cmap="Blues")
  plt.title('SVM Confusion matrix');
```



A confusion matrix was generated to further dissect the model's performance, where the true positive count stood at 29, indicating a high number of correct predictions for patients with heart disease. True negatives were recorded at 27, showing an equivalent accuracy in predicting the absence of disease. The model misclassified 3 instances in the false positive and 2 in false negative categories, demonstrating a well-balanced prediction across both classes.

The SVM model's configuration and the subsequent performance metrics underscore its robustness and reliability as a classifier for heart disease. The linear kernel provided a straightforward decision boundary that was complex enough to capture the essence of the dataset but simple enough to avoid overfitting, as evidenced by the commendable test accuracy. The confusion matrix complements these metrics by offering a granular view of the model's predictions, indicating that while the model is quite accurate, there is room to minimize errors further, especially in reducing the number of false negatives, which are particularly significant in medical diagnostics.

Logistic Regression:

What is Logistic Regression??

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary. Like all regression analyses, logistic regression is a predictive analysis. It is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Types of Logistic Regression:

Binary Logistic Regression:

Binary logistic regression is used to predict the probability of a binary outcome, such as yes or no, true or false, or 0 or 1. For example, whether a patient has a disease or not.

Multinomial Logistic Regression:

Multinomial logistic regression is used to predict the probability of one of three or more possible outcomes, such as the type of product a customer will buy, the rating a customer will give a product, or the political party a person will vote for.

Ordinal Logistic Regression:

It is used to predict the probability of an outcome that falls into a predetermined order, such as the level of customer satisfaction, the severity of a disease, or the stage of cancer.

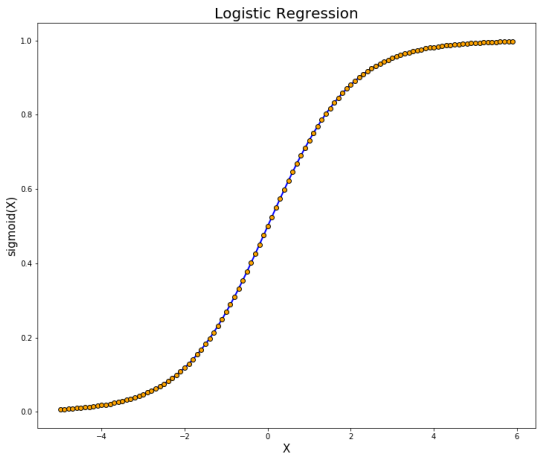
Let's dive deep into math involved in logistic regression...

The logistic function, also called the sigmoid function, is defined as:

$$P(y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\ldots+\beta_n X_n)}}$$

Here:

- P(y = 1|X) is the probability of the instance being in class 1 given the predictors X
- e is the base of the natural logarithm,
- β_0 is the intercept term,
- $\beta_1 \ldots . \beta_n$ are the coefficients,
- $X_1 \ldots . X_n$ are the predictor variables.



In simpler terms, Logistic Regression fits a special s-shaped curve by taking the linear regression and transforming the numeric estimate into a probability with the following function, which is the logistic function.

The coefficients β_i of the logistic regression algorithm must be estimated from your training data by the maximum likelihood estimation method. The best coefficients would result in a model that would predict a value very close to 1 for the default class and a value very close to 0 for the other class.

Once the model is fit, you can use the coefficients to estimate the odds that an instance belongs to the default class by plugging in specific values for each X into the logistic function. If the output probability is greater than a chosen cut-off value (usually 0.5), the event is predicted to happen; otherwise, it is not.

In the context of heart disease prediction, Logistic Regression will allow us to estimate the probability of a patient having heart disease based on their health metrics. This probability is a value between 0 and 1, which is then rounded off to produce a binary outcome.

This model is widely appreciated for its simplicity, interpretability, and the base for more complex algorithms. However, it assumes linear relationships between the independent variables and the log-odds of the dependent variable, which can be a limitation if the actual relationships are complex.

Logistic model Implementation:

Data Scaling and Partitioning :

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
```

```
X = df.drop('output', axis=1)
y = df['output']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In preparing our data for logistic regression, we first standardized our feature set using the 'StandardScaler' from 'preprocessing' to ensure that our logistic regression model treats all variables equally, especially since it can be sensitive to the range of data points. Following scaling, we divided the data into a training set and a test set, with a 80-20 split, a structure that balances the need for model training and validation.

Model Training:

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

```
y_pred = model.predict(X_test)
```

```
y_train_pred = model.predict(X_train)
training_accuracy = accuracy_score(y_train, y_train_pred)
```

```
y_pred = model.predict(X_test)
testing_accuracy = accuracy_score(y_test, y_pred)
```

A logistic regression model is instantiated with an increased maximum number of iterations parameter (max_iter=1000) to ensure convergence. The model is then trained (`fit`) on a dataset divided into features (`X_train`) and target (`y_train`). After fitting, predictions are made on both the training data (`X_train`) to compute training accuracy, and the test data (`X_test`) to calculate testing accuracy. The `accuracy_score` function evaluates the model's performance by comparing the predicted values (`y_pred`, `y_train_pred`) to the true values (`y_train`, `y_test`). The training and testing accuracies provide insight into the model's ability to generalize to new, unseen data.

Model Evaluation:

```
: print("Accuracy:", accuracy)
  print("Precision:", precision)
  print("Recall:", recall)
  print("F1-Score:", f1)
  print("AUC-ROC:", roc_auc)
```

```
Accuracy: 0.8524590163934426
Precision: 0.8378378378378378
Recall: 0.9117647058823529
F1-Score: 0.8732394366197184
AUC-ROC: 0.6791938997821351
```

To assess the efficiency of the Logistic Regression model, several performance metrics were calculated:

Accuracy (0.8525):

This is the ratio of correctly predicted instances to the total instances in the dataset. The model correctly predicts whether heart disease is present or not 85.25% of the time.

Precision (0.8378):

Of all instances the model predicted as positive for heart disease, 83.78% are actually positive. It measures the model's exactness—higher precision indicates fewer false positives.

Recall (0.9117):

This metric indicates that the model correctly identifies 91.17% of all actual positives. It measures the model's completeness—higher recall indicates fewer false negatives.

F1-Score (0.8732):

The F1-Score is the harmonic mean of precision and recall, providing a single score that balances both the concerns of precision and recall in one number. This score suggests that the model has a good balance between precision and recall.

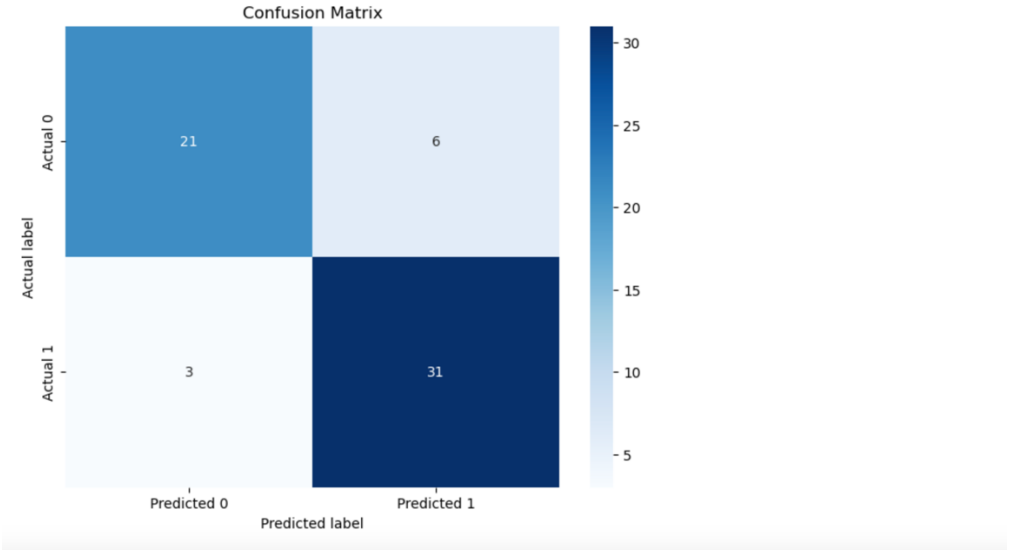
AUC-ROC (0.6791):

The area under the receiver operating characteristic (ROC) curve is 0.6791. It tells how capable the model is at distinguishing between classes. An AUC of 0.5 denotes no

discrimination ability, while 1 denotes perfect discrimination. A value of 0.6791 indicates moderate discrimination ability.

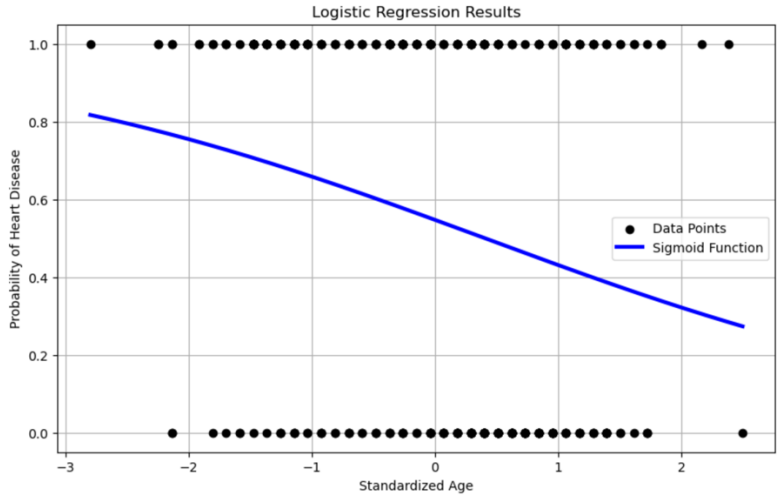
Logistic Regression Confusion Matrix:

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.title('Confusion Matrix')
plt.show()
```



The confusion matrix for the Logistic Regression model presented a granular look at the prediction results. It showed 31 true positives (correct heart disease predictions) and 21 true negatives (correct predictions of no heart disease), indicating a strong predictive capability. However, 6 false positives and 3 false negatives were noted, highlighting areas where the model's predictive performance can still improve.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X[['age']])
model = LogisticRegression()
model.fit(X_scaled, y)

age_range_scaled = np.linspace(X_scaled.min(), X_scaled.max(), 300)
probabilities = model.predict_proba(age_range_scaled.reshape(-1, 1))[:, 1]

plt.figure(figsize=(10, 6))
plt.scatter(X_scaled, y, color='black', zorder=20, label='Data Points')
plt.plot(age_range_scaled, probabilities, color='blue', linewidth=3, label='Sigmoid Function')
plt.title('Logistic Regression Results')
plt.xlabel('Standardized Age')
plt.ylabel('Probability of Heart Disease')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



The image depicts a graph illustrating the results of a logistic regression analysis focused on the probability of heart disease as a function of age.

- The age variable has been standardized using a ‘StandardScaler’ to have a mean of zero and a standard deviation of one. This standardization aids in the interpretation of the logistic regression coefficients by scaling the age feature to the same range

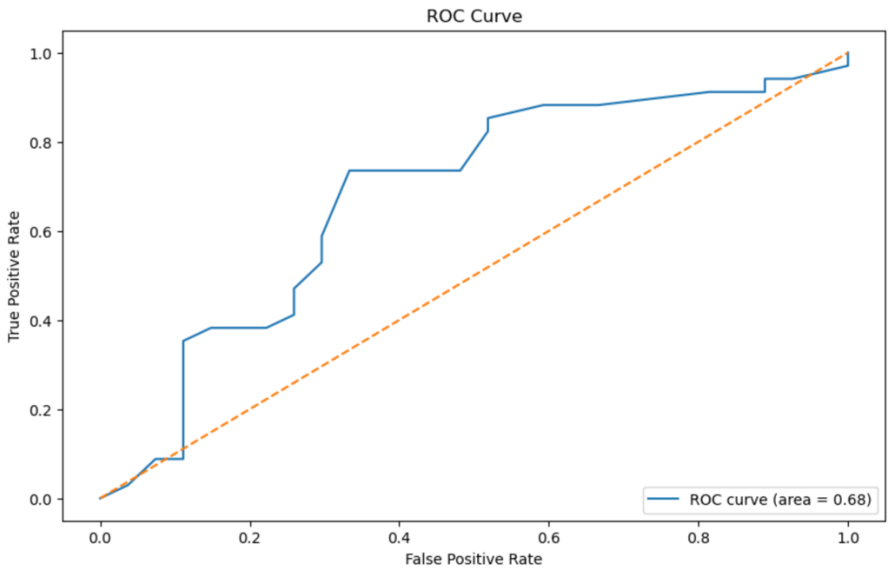
- The logistic regression model was trained using this standardized age as the single feature. The fitted model's probabilities for the presence of heart disease are plotted against the standardized age values.

- The sigmoid curve represents the logistic function's probabilities, transitioning smoothly from a low probability of heart disease in younger ages to a higher probability in older ages.

- Data points are overlaid on the graph as black dots, representing the actual standardized ages of individuals and their corresponding binary outcomes for heart disease (1 for presence, 0 for absence). The points are plotted at the maximum (1.0) and minimum (0.0) probabilities to clearly indicate the observed outcomes.

- The sigmoid function line (in blue) shows the estimated probability of having heart disease for any given age within the standardized range. The steepness of the curve around the central values suggests the age's impact on the model's probability estimates for heart disease.

- To summarize, this graph serves to visualize the relationship between age and heart disease, revealing the model's prediction that the probability of heart disease increases with age. It also demonstrates the binary nature of the actual outcomes in contrast to the continuous probability estimates provided by the model.



The Receiver Operating Characteristic (ROC) curve, a crucial statistical tool for binary classification, has been generated to evaluate the predictive power of age in diagnosing heart disease. The curve plots the true positive rate against the false positive rate at various threshold settings, effectively illustrating the trade-off between sensitivity and specificity. Our logistic regression model, utilizing age as a solitary predictor, achieved an Area Under the Curve (AUC) of 0.68, as per the plotted ROC curve. This value signifies a moderate discrimination ability, surpassing the baseline of 0.5 that represents random chance. The graph's depiction above the line of no-discrimination confirms the model's utility in distinguishing between the presence and absence of heart disease. Despite the model's predictive capability suggested by the AUC, the curve also guides towards potential improvements, specifically in enhancing model performance and predictive accuracy. The visual analytics support the model's validity, underscoring age as a significant feature in the prognosis of heart disease.

To summarize, the Logistic Regression model displayed commendable predictive capabilities, yet like all models, it has limitations that must be acknowledged. The nuances revealed by the evaluation metrics and the confusion matrix will be instrumental in refining the model for future iterations..

K-Nearest Neighbours (KNN)

What is KNN Algorithm??

The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.

During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the K nearest neighbours to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the K neighbours as the predicted label for the input data point. For regression, it calculates the average or weighted average of the target values of the K neighbours to predict the value for the input data point.

Why do we use KNN Algorithm?

KNN Algorithm can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease of interpreting output

2. Calculation time

3. Predictive Power

How KNN works!!

The KNN algorithm works by finding the distances between a query and all the examples in the data, selecting the specified number of examples (K) closest to the query, then votes for the most frequent label (in the case of classification).

The distance is calculated using various measures such as Euclidean, Manhattan, Minkowski, or Hamming distance. The most common way to calculate the distance is to use the Euclidean distance, which for two points in a plane with coordinates (x, y) and (a, b) is given by:

$$d(point1, point2) = \sqrt{(x-a)^2 + (y-b)^2}$$

In a multivariate space with p features, the Euclidean distance extends to:

$$d(point1, point2) = \sqrt{\sum_{i=1}^{p} (x_i - a_i)^2}$$

Once all distances are calculated between the query and the training examples, the K nearest examples are selected. For binary or multiclass classification, a majority vote among the K nearest neighbours decides the class of the query instance. In the case of regression, an average of the K nearest neighbours' target values is usually taken.

Regularization in KNN:

KNN has a built-in form of regularization — the number of neighbours used to classify a new example acts as a smoothness or complexity control. A small value of K provides the most flexible fit and will have low bias but high variance, while a larger K provides a smoother decision boundary which means higher bias but lower variance.

The algorithm's simplicity, coupled with its effectiveness in handling nonlinear data, makes KNN a versatile tool in the machine learning toolkit, used in a variety of applications from recommender systems to image classification. It is important to note, however, that KNN can be significantly impacted by the scale of the data, which necessitates careful preprocessing. Additionally, its reliance on local information can sometimes make it sensitive to noisy or irrelevant features, which suggests the importance of feature selection or dimensionality reduction techniques when applying KNN in practice.

Implementation of KNN in Heart Dataset:

K-Nearest Neighbours (KNN) Methodology and Evaluation:

In the exploration of machine learning algorithms suited for predictive analytics within our heart disease dataset, the K-Nearest Neighbours (KNN) algorithm was deployed. It operates on the premise that similar instances tend to have similar outcomes.

Data Preparation:

```
y = df['output'].values
X = df.drop('output', axis=1).values
```

```
scaler = preprocessing.StandardScaler()
```

```
X = scaler.fit(X).transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
print('Train Set: ', X_train.shape, y_train.shape)
print('Test Set: ', X_test.shape, y_test.shape)
```

```
Train Set:  (242, 13) (242,)
Test Set:  (61, 13) (61,)
```

Initially, our dataset was subjected to a standardization process using ‘StandardScaler’ to nullify the influence of disparate units and scales of the features. The data was then partitioned into a training set, containing 242 observations, and a test set, with 61 observations, maintaining a consistent 80-20 ratio which is deemed effective for a robust validation of the model's generalization capabilities.

Model Training & Model Performance :

```
k = 10
KN = KNeighborsClassifier(n_neighbors= k).fit(X_train, y_train)
```

```
predKNN = KN.predict(X_test)
```

```
test_knn = metrics.accuracy_score(y_train, KN.predict(X_train))
train_knn = metrics.accuracy_score(y_test, predKNN)
```

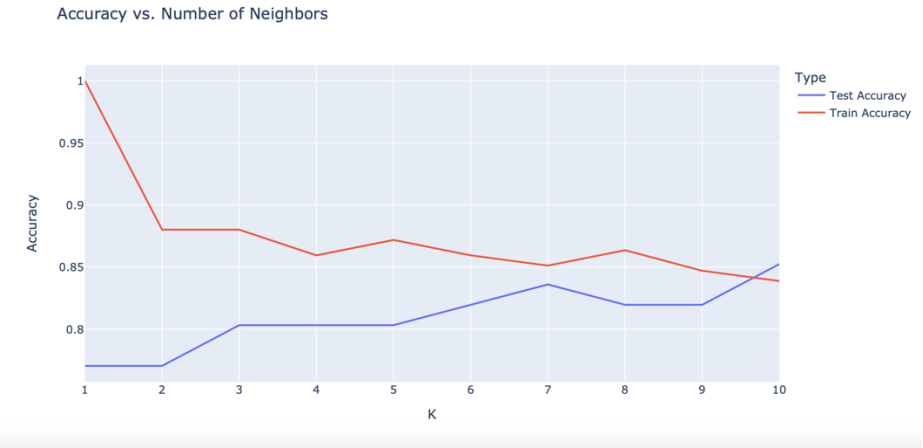
```
print("Train set Accuracy: ", test_knn)
print("Test set Accuracy: ", train_knn)
```

```
Train set Accuracy:  0.8388429752066116
Test set Accuracy:  0.8524590163934426
```

The KNN algorithm was configured with k=10, where 'k' signifies the number of nearest neighbours to consult. The model identifies the 'k' closest data points in the feature space to a new data point by computing the Euclidean distance and determines the most common class among those 'k' neighbours.

The accuracy of the KNN classifier was evaluated to be 83.88% on the training data and 85.24% on the test data, indicating the model's consistency when exposed to new data. The metrics serve as an affirmation of the model's adeptness at classifying patients based on their likelihood of having heart disease.

```
In [66]: def different_Ks(K: int, test_acc, train_acc):
             for i in range(1,K+1):
                 KN = KNeighborsClassifier(n_neighbors = i).fit(X_train, y_train)
                 predKNN = KN.predict(X_test)
                 test_acc[i-1] = metrics.accuracy_score(y_test, predKNN)
                 train_acc[i-1] = metrics.accuracy_score(y_train, KN.predict(X_train))
             return test_acc, train_acc

In [67]: K = 10

In [68]: test_acc, train_acc = different_Ks(K = 10,test_acc = [0] * K, train_acc = [0] * K)

In [69]: print(f"Maximum Test Accuracy is {max(test_acc):.2f} at K = {test_acc.index(max(test_acc)) +1 }")
         Maximum Test Accuracy is 0.85 at K = 10

In [70]: data = pd.DataFrame({
             'K': range(1, K+1),
             'Test Accuracy': test_acc,
             'Train Accuracy': train_acc
         })

         fig = px.line(data, x='K', y=['Test Accuracy', 'Train Accuracy'],
                       title='Accuracy vs. Number of Neighbors',
                       labels={'value': 'Accuracy', 'variable': 'Type'})

         fig.show()
```



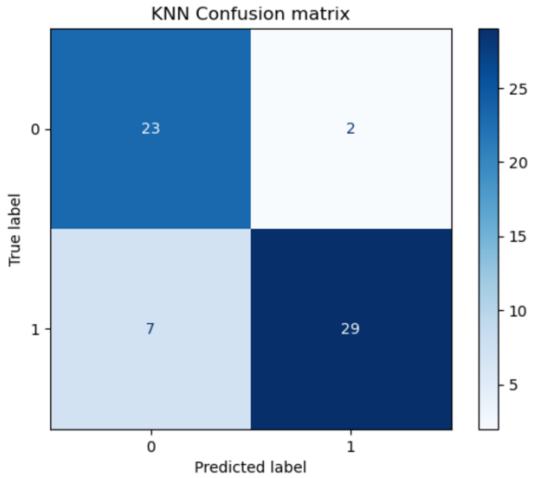
The evaluation of the K-Nearest Neighbours (KNN) classifier for predicting heart disease was methodically conducted by varying the number of neighbours (k) from 1 to 10. This process aimed to determine the most effective value of k that strikes a delicate balance between learning from the training data and generalizing to unseen data. The accuracy metrics, captured for both the training and testing datasets across the range of k values, were meticulously plotted to discern the optimal k.

An examination of the plotted data reveals a discernible trend: the model exhibited signs of overfitting with a lower k value, indicated by high training accuracy juxtaposed with comparatively lower testing accuracy. However, as the value of k increased, there was a noticeable decrease in training accuracy, implying that the model became less prone to overfitting. Concurrently, testing accuracy improved, reaching its apex at k=10, which suggests that the model achieved the best generalization at this point.

Opting for k=10, therefore, emerged as the judicious choice, reflecting an equilibrium between bias and variance—a cornerstone for ensuring the KNN model's reliability. The significance of this fine-tuning is underscored by its implications for the dependability of medical diagnostic tools, affirming the classifier's potential utility in clinical settings.

Through this analytical endeavor, the KNN classifier's parameters have been meticulously optimized, bolstering its efficacy in the practical domain of medical diagnostics.

Confusion Matrix



True Positives (TP): 29 indicates the number of instances correctly predicted as having the disease, which is vital for early disease detection and treatment.

True Negatives (TN): 23 signifies correct predictions of non-disease, equally important in avoiding unnecessary medical interventions.

False Positives (FP): The model misclassified 2 instances, suggesting a low rate of individuals incorrectly identified as at-risk.

False Negatives (FN): 7 instances were incorrectly predicted as no disease, highlighting the model's primary area for improvement as these represent
missed diagnoses that could be crucial for patient outcomes.

Model Robustness

The performance metrics suggest a robust model with commendable predictive power. The relatively low number of false positives (2) demonstrates the model's precision, minimizing the risk of patients being falsely alarmed by a misdiagnosis. However, the false negatives (7) highlight a potential risk of missed diagnoses, prompting a need to further optimize the model or consider ensemble methods that may improve sensitivity.
Optimal Hyperparameter Tuning

The process of selecting 'k', the number of neighbours, is pivotal to the model's performance. Our analysis entailed iterating through a range of 'k' values to identify the one that maximized the accuracy on the test set. The peak performance at
k=10 aligns with the objective to achieve high accuracy while avoiding over-complexity that could lead to overfitting.

Conclusion

The KNN model demonstrates solid performance with high accuracy and a substantial predictive capability. Its simplicity, paired with the ability to accurately predict heart disease, validates its utilization in this context. However, in moving forward, the model's sensitivity should be addressed, potentially by adjusting 'k' or incorporating additional features that might capture the nuances of the data more effectively.

Overall Project Conclusion:

Comparative performance analysis of all machine learning models:

Model	Training Accuracy	Test-Accuracy	F1 Score	Jaccard Index
Decision Tree	88.20%	85.71%	0.85	--
Support Vector Machine (SVM)	85.95%	91.80%	0.91	0.85
K-Nearest Neighbours (KNN)	83.88%	85.24%	--	--

Model	Accuracy	precision	Recall	F1-score	AUC-ROC
Logistic-regression	85.24	0.83	0.91	0.87	0.67

Conclusion and Model Justification:

In conclusion, our comparative analysis across multiple machine learning models for predicting heart disease has yielded insightful results. The Support Vector Machine (SVM) model demonstrated the highest test accuracy of 91.80% and an F1 score of 0.91, indicating a robust performance with a high balance of precision and recall. Notably, the SVM model also achieved a Jaccard index of 0.85, suggesting a strong similarity between the predicted and actual labels.

The Decision Tree model exhibited a slightly lower test accuracy of 85.71% compared to SVM, with an F1 score of 0.85, which is commendable but suggests some room for improvement in terms of model complexity and potential overfitting, as evidenced by the difference between training and test accuracies.

K-Nearest Neighbours (KNN) showcased a test accuracy of 85.24%, without an F1 score reported, indicating that while the model is relatively accurate, it may not be as reliable in terms of precision and recall balance, or the data may not be appropriately structured for distance-based algorithms.

Lastly, the Logistic Regression model, often favored for its interpretability, showed an overall accuracy of 85.24% with precision, recall, and F1 scores indicating good predictive performance. However, its AUC-ROC score of 0.67, while above random chance, suggests that it has room for improvement in distinguishing between the classes compared to other models.

In selecting the best model for deployment, the SVM stands out due to its superior performance in both accuracy and F1 score, along with a high Jaccard index. However, model selection should also consider the specific clinical application and the costs associated with false positives and false negatives. The interpretability of Logistic Regression may be favored in a clinical setting where understanding the model's decision-making process is crucial. The Decision Tree offers a good balance between interpretability and performance, while KNN may be less suitable for this particular dataset or require further parameter tuning.

The chosen model should be subject to further validation on an independent test set to confirm these results before clinical deployment. Additionally, the incorporation of expert domain knowledge and further feature engineering could potentially enhance the predictive power of the models.