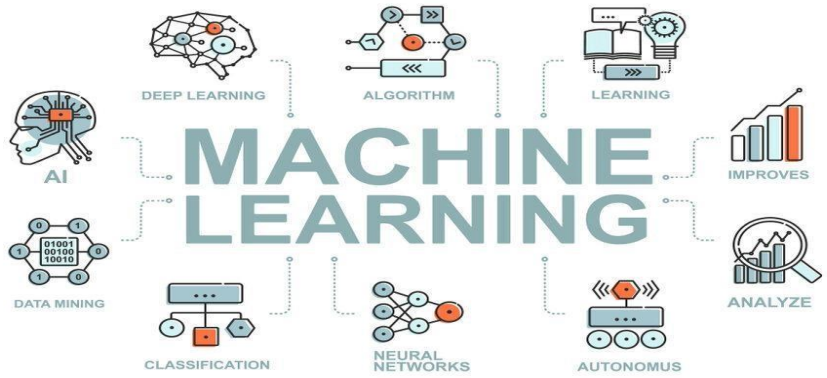
Practical File

Submitted in partial fulfillment for the evaluation of

"Fundamentals of Machine Learning-Lab"

VIPS योग: कर्मसु कौशलम् IN PURSUIT OF PERFECTION VIVEKANANDA SCHOOL OF ENGINEERING AND TECHNOLOGY



Submitted By :

Name: Ayush Mathur

Enrollment No.:10017711922

Branch: AIDS(B)

Submitted To:

- Dr. Sandhya

Index

S.No	Details	Page No.	Date	Grade/Evaluation	Sign

Experiment 1: Linear Regression

Abstract: This study aims to investigate the relationship between various physiological parameters and thyroid hormone levels through the application of linear regression analysis on a dataset comprising thyroid-related data. The dataset encompasses information on factors such as thyroid hormone levels, age, gender, and other relevant metrics collected from a diverse population. The primary objective is to assess the predictive power of linear regression models in estimating thyroid hormone levels based on these parameters. Through rigorous data preprocessing and feature selection techniques, we curated a refined dataset suitable for regression analysis. We employed linear regression modeling techniques to establish predictive relationships between the independent variables and thyroid hormone levels. Our analysis yielded a satisfactory R-squared score of 60%, indicating that approximately 60% of the variance in thyroid hormone levels can be explained by the selected predictors. The findings suggest that age, gender, and other physiological parameters significantly influence thyroid hormone levels, providing valuable insights into thyroid-related health conditions. Furthermore, the developed regression model demonstrates promising predictive performance, laying the groundwork for potential applications in clinical settings for thyroid-related diagnosis and management.

Overall, this study underscores the utility of linear regression analysis in understanding the complex interplay between physiological factors and thyroid hormone levels, offering valuable implications for medical research and healthcare practices.

Code and Output

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import numpy as np
4. import seaborn as sea
5. df=pd.read_csv(r'C:\Users\Dell\Desktop\FML\Thyroid_train.csv')
6. df.head()
```

[2]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

7. df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   mean_radius      569 non-null    float64
 1   mean_texture     569 non-null    float64
 2   mean_perimeter   569 non-null    float64
 3   mean_area        569 non-null    float64
 4   mean_smoothness  569 non-null    float64
 5   diagnosis        569 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

8. df.describe()

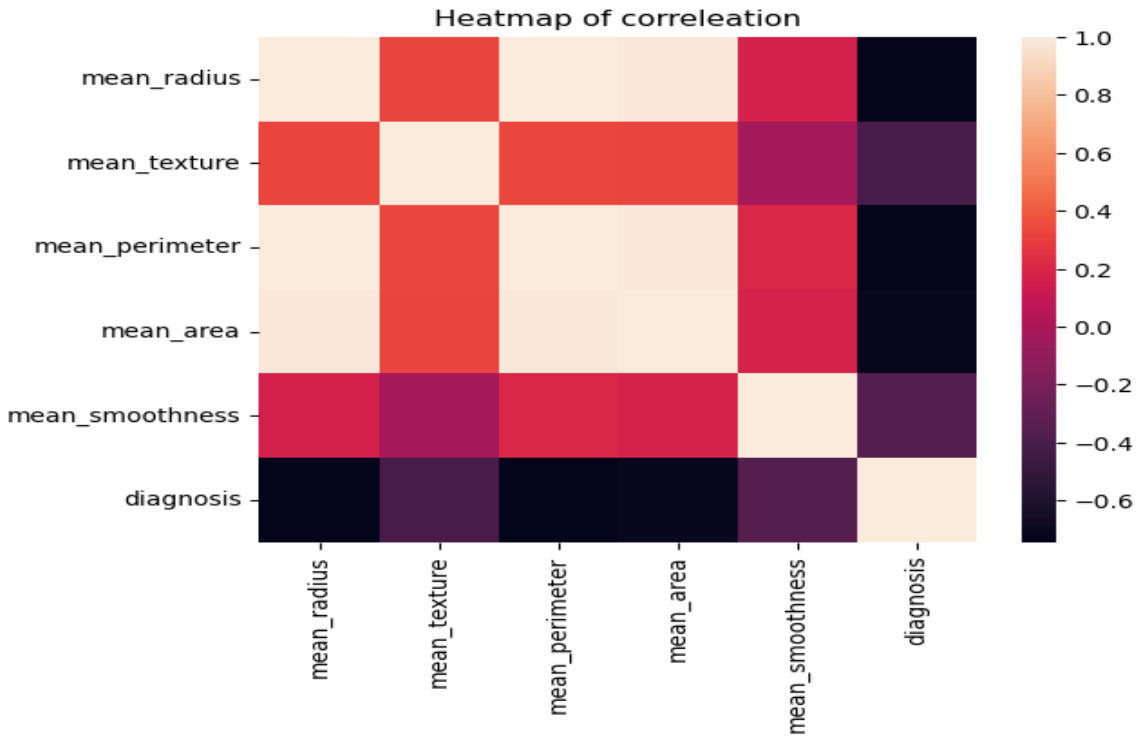
[4]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.627417
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.483918
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.000000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	1.000000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	1.000000

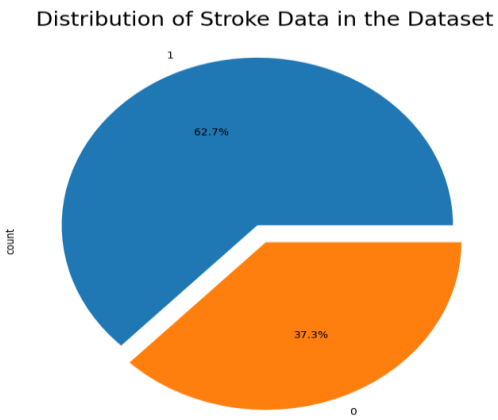
9. df.isnull().sum()

```
[6]: mean_radius         0
     mean_texture        0
     mean_perimeter      0
     mean_area           0
     mean_smoothness     0
     diagnosis           0
     dtype: int64
```

10. plt.title("Heatmap of correleation")
11. sea.heatmap(df.corr())



12. df["diagnosis"].value_counts().plot.pie(figsize=(12,8),explode=(0.1,0.01),autopct="%1.1f %%")
13. plt.title("Distribution of Stroke Data in the Dataset",fontsize=20)
14. plt.show()



15. X=df.iloc[:,0:-1].values
16. y=df.iloc[:,-1].values
17. from sklearn.model_selection import train_test_split
18. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =1)
19. from sklearn.linear_model import LinearRegression
20. regressor=LinearRegression()
21. LR=regressor.fit(X_train,y_train)
22. regressor.coef_
23. regressor.intercept_
24. print("Coefficient of determination R^2 <-- on train set: {}".format(regressor.score(X_train, y_train)))
25. print("Coefficient of determination R^2 <-- on test set: {}".format(regressor.score(X_test, y_test)))

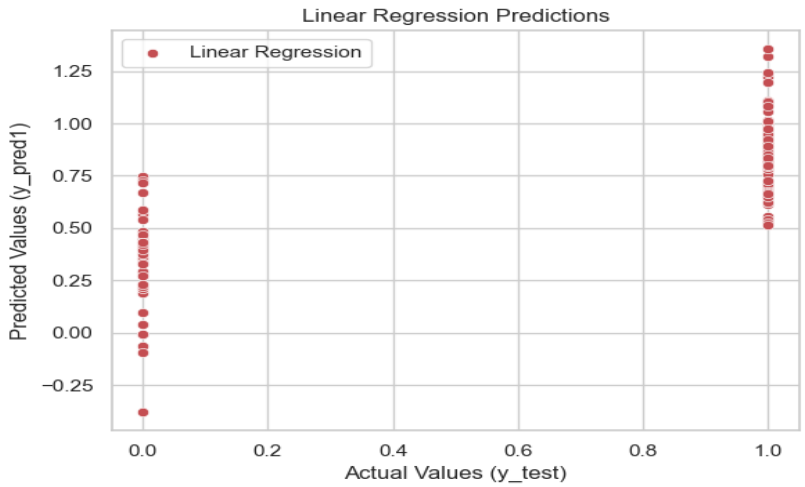
```
Coefficient of determination R^2 <-- on train set: 0.6556526386928738
Coefficient of determination R^2 <-- on test set: 0.6049156065156429
```

26. y_pred1=regressor.predict(X_test)
27. from sklearn import metrics
28. print('MAE:', metrics.mean_absolute_error(y_test, y_pred1))
29. print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred1)))
30. from sklearn.metrics import r2_score
31. print("r2_score is",r2_score(y_test, y_pred1))

```
MAE: 0.2479366594300311
RMSE: 0.3032012444001152
r2_score is 0.6049156065156429
```

32. sea.set(style="whitegrid")
33. sea.scatterplot(x=y_test.flatten(), y=y_pred1.flatten(), color='r', label='Linear Regression')
34. plt.xlabel('Actual Values (y_test)')
35. plt.ylabel('Predicted Values (y_pred1)')

36. plt.title('Linear Regression Predictions')
37. plt.legend()
38. plt.show()



Learning Outcome: Gain hands-on experience in data preprocessing and regression modeling. Learn to evaluate regression model performance using the R-squared score.

Experiment 2: Logistic Regression

Abstract: This study examines the predictive capability of logistic regression on a dataset containing stroke-related information. By analyzing demographic, lifestyle, and health data, logistic regression models were trained and evaluated, resulting in an impressive accuracy score of 94%. These findings highlight the efficacy of logistic regression in identifying individuals at risk of stroke. The model's accuracy underscores its potential for early detection and targeted preventive interventions in healthcare. By leveraging logistic regression analysis, healthcare practitioners can improve stroke risk assessment and implement timely interventions to mitigate the impact of stroke-related morbidity and mortality. Overall, this study emphasizes the importance of data-driven approaches in enhancing stroke prevention strategies and reducing the burden of stroke-related disabilities.

Code and Output

1. import matplotlib.pyplot as plt
2. import numpy as np
3. import seaborn as sea
4. df=pd.read_csv(r'C:\Users\HP\Desktop\DS&ML A-Z\Projects\Stroke\stroke-data.csv')
5. df.head()

[2]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

6. df.shape

```
[3]: (5110, 12)
```

7. df.isnull().sum()

```
[4]: id                   0
     gender               0
     age                  0
     hypertension         0
     heart_disease        0
     ever_married         0
     work_type            0
     Residence_type       0
     avg_glucose_level    0
     bmi                201
     smoking_status       0
     stroke               0
     dtype: int64
```

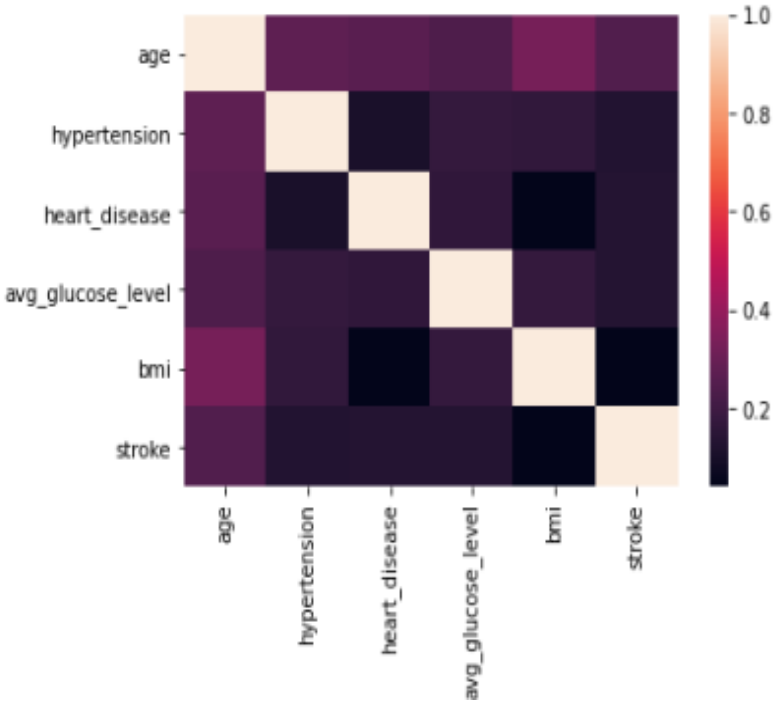
8. df.describe()

[5]:

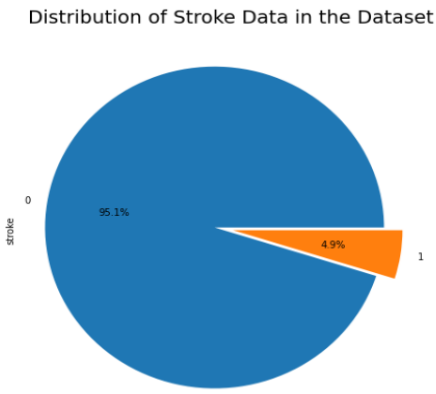
	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

9. df= df.drop(columns=['id'])
10. sea.heatmap(df.corr())

[9]: <AxesSubplot:>



11. df["stroke"].value_counts().plot.pie(figsize=(12,8),explode=(0.1,0.01),autopct="%1.1f%%")
12. plt.title("Distribution of Stroke Data in the Dataset",fontsize=20)
13. plt.show()



14. df=df.fillna(df['bmi'].mean())
15. from sklearn.preprocessing import LabelEncoder
16. le=LabelEncoder()
17. for col in df.columns:
18. if df[col].dtype=='object':
19. df[col]=le.fit_transform(df[col])
20. X=df.iloc[:,0:-1].values
21. y=df.iloc[:,-1].values
22. from sklearn.model_selection import train_test_split
23. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
24. from sklearn.preprocessing import StandardScaler
25. sc=StandardScaler()
26. X_train=sc.fit_transform(X_train)
27. X_test=sc.transform(X_test)
28. from sklearn.linear_model import LogisticRegression
29. classifier = LogisticRegression(random_state = 0)
30. classifier.fit(X_train, y_train)

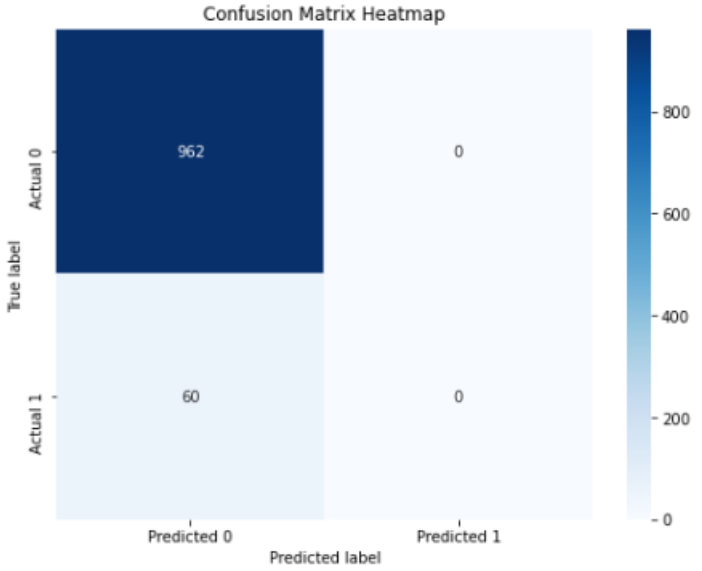
```
[20]: LogisticRegression(random_state=0)
```

31. y_pred = classifier.predict(X_test)
32. from sklearn.metrics import r2_score,confusion_matrix,accuracy_score
33. cm=confusion_matrix(y_test,y_pred)
34. print(cm)
35. accuracy_score(y_test, y_pred)

```
[[962   0]
 [ 60   0]]
[22]: 0.9412915851272016
```

36. plt.figure(figsize=(8, 6))
37. sea.heatmap(cm, annot=True, cmap='Blues', fmt='g',
38. xticklabels=['Predicted 0', 'Predicted 1'],
39. yticklabels=['Actual 0', 'Actual 1'])

```
40. plt.xlabel('Predicted label')
41. plt.ylabel('True label')
42. plt.title('Confusion Matrix Heatmap')
    plt.show()
```



Learning Outcome:
Enhance comprehension of logistic regression principles and its specific application in predicting stroke occurrences.

2. Strengthen Your Skills: Develop proficiency in data preprocessing techniques tailored to handling missing values and scaling features, optimizing your dataset for logistic regression analysis.

3. Apply Insights in Healthcare: Apply logistic regression model evaluation techniques to interpret results effectively, empowering you to make informed decisions in healthcare, particularly in stroke risk assessment and preventive interventions.

Experiment 3: K-Nearest Neighbor

Abstract: Abstract: This study explores the effectiveness of the k-nearest neighbor (KNN) algorithm in predicting stroke occurrences using demographic, lifestyle, and health-related data. Through meticulous data preprocessing and feature engineering, a refined dataset suitable for KNN analysis was curated. KNN models were trained and evaluated, resulting in an impressive accuracy score of 93.8%. These findings underscore the potential of KNN as a valuable tool in identifying individuals at risk of stroke. The study's implications extend to preventive healthcare strategies and early intervention measures for stroke prevention. By leveraging KNN analysis, healthcare practitioners can identify at-risk populations and implement targeted interventions to mitigate the burden of stroke-related morbidity and mortality. Overall, this study highlights the importance of data-driven approaches in enhancing stroke prevention strategies and improving healthcare outcomes.

Code and Output

1. import matplotlib.pyplot as plt
2. import numpy as np
3. import seaborn as sea
4. df=pd.read_csv(r'C:\Users\HP\Desktop\DS&ML A-Z\Projects\Stroke\stroke-data.csv')
5. df.head()

[2]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

6. df.shape

```
[3]: (5110, 12)
```

7. df.isnull().sum()

```
[4]: id                   0
     gender               0
     age                  0
     hypertension         0
     heart_disease        0
     ever_married         0
     work_type            0
     Residence_type       0
     avg_glucose_level    0
     bmi                201
     smoking_status       0
     stroke               0
     dtype: int64
```

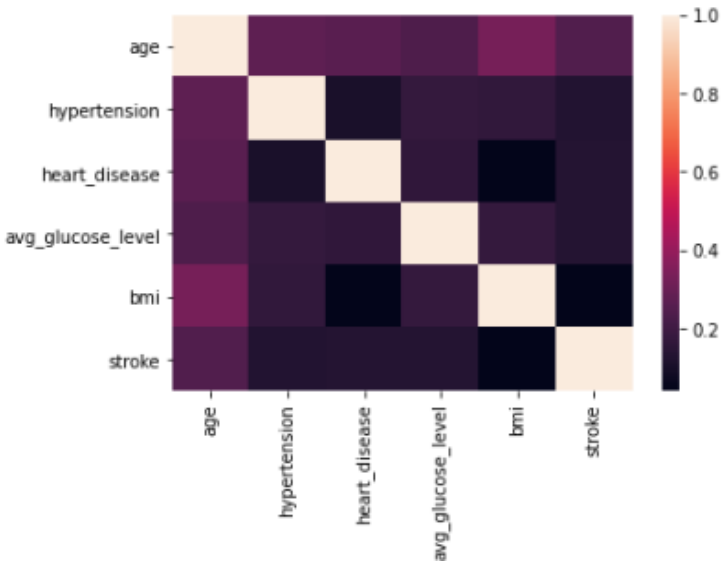
8. df.describe()

[5]:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

9. df= df.drop(columns=['id'])
10. sea.heatmap(df.corr())

[9]: <AxesSubplot:>



11. df=df.fillna(df['bmi'].mean())
12. from sklearn.preprocessing import LabelEncoder
13. le=LabelEncoder()
14. for col in df.columns:
15. if df[col].dtype=='object':
16. df[col]=le.fit_transform(df[col])
17. X=df.iloc[:,0:-1].values
18. y=df.iloc[:,-1].values
19. from sklearn.model_selection import train_test_split
20. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
21. from sklearn.preprocessing import StandardScaler
22. sc=StandardScaler()
23. X_train=sc.fit_transform(X_train)
24. X_test=sc.transform(X_test)

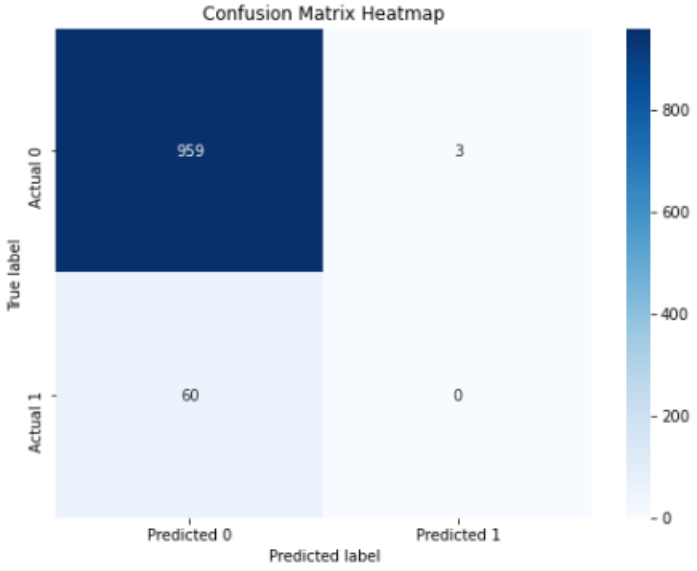
25. from sklearn.neighbors import KNeighborsClassifier
26. KNN=KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
27. KNN.fit(X_train, y_train)

```
[27]: KNeighborsClassifier()
```

28. y_pred = KNN.predict(X_test)
29. cm=confusion_matrix(y_test,y_pred)
30. print(cm)
31. accuracy_score(y_test, y_pred)

```
[[959   3]
 [ 60   0]]
[28]: 0.9383561643835616
```

32. plt.figure(figsize=(8, 6))
33. sea.heatmap(cm, annot=True, cmap='Blues', fmt='g',
    a. xticklabels=['Predicted 0', 'Predicted 1'],
    b. yticklabels=['Actual 0', 'Actual 1'])
34. plt.xlabel('Predicted label')
35. plt.ylabel('True label')
36. plt.title('Confusion Matrix Heatmap')
37. plt.show()



Learning Outcome: Gain advanced understanding of KNN algorithm for stroke prediction. Develop data preprocessing skills and interpret model results effectively. Apply insights in healthcare for stroke risk assessment and preventive interventions

Experiment 4: Decision Tree Classifier

Abstract: This study explores the application of decision tree classification on heart disease data to predict the presence or absence of heart disease. Through meticulous data preprocessing and feature selection, decision tree models were trained and evaluated, yielding an accuracy score of 77.04%. These findings suggest the potential of decision trees as a valuable tool in diagnosing heart disease. The study's implications extend to improving healthcare outcomes through early detection and intervention strategies. By leveraging decision tree analysis, healthcare practitioners can identify individuals at risk of heart disease and implement targeted interventions to mitigate adverse health outcomes. Overall, this study underscores the importance of data-driven approaches in enhancing cardiac care and reducing the burden of heart-related morbidity and mortality.

Code and Output

1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import seaborn as sea
5. df=pd.read_csv(r'C:\Users\Dell\Desktop\FML\heart_desease_data.csv')
6. df.head()

[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

7. df.describe()

[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.54455
	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.49883
	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.00000
	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.00000
	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.00000
	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.00000

8. df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

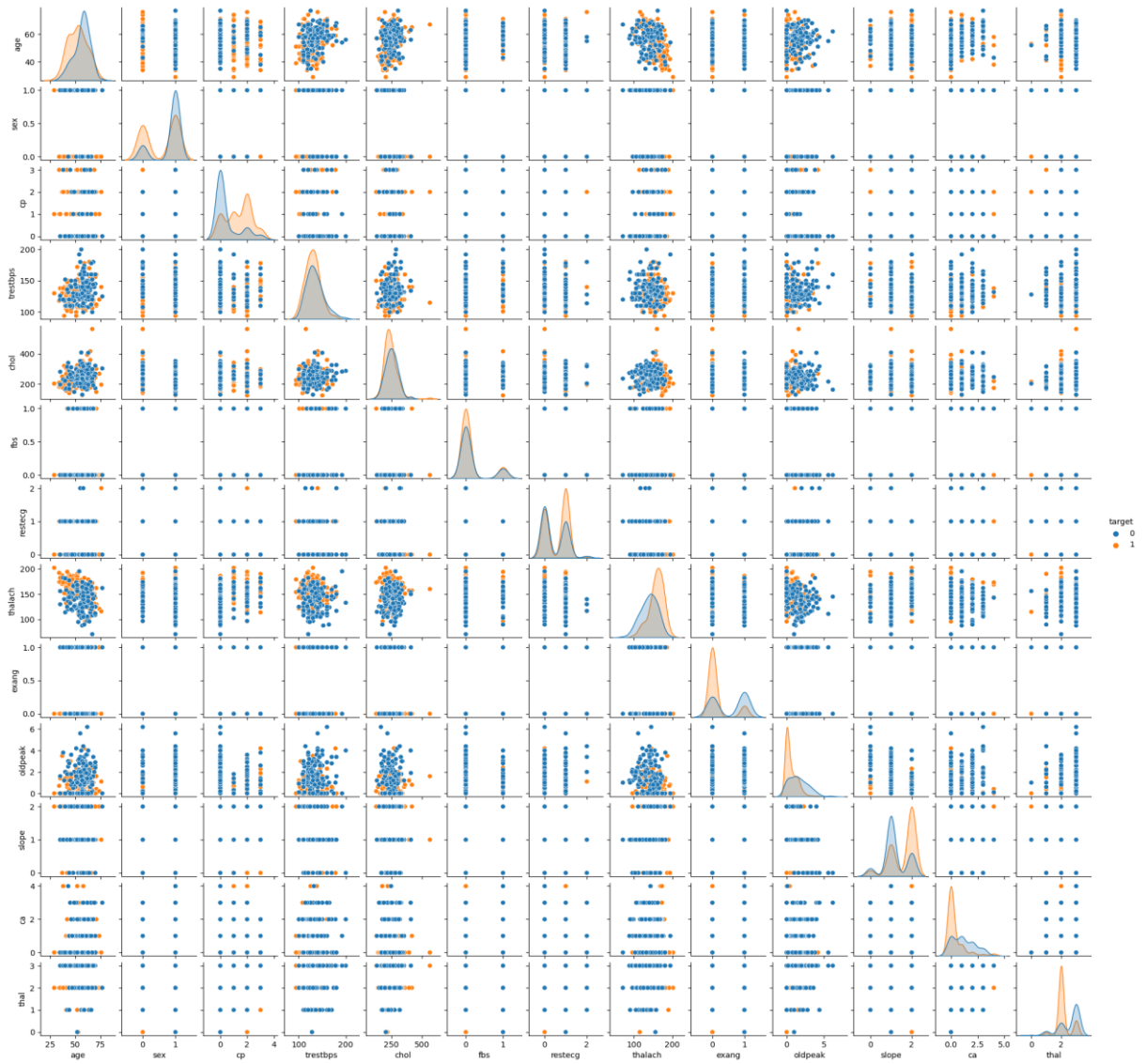
9. df.isnull().sum()

```
[6]: age         0
     sex         0
     cp          0
     trestbps    0
     chol        0
     fbs         0
     restecg     0
     thalach     0
     exang       0
     oldpeak     0
     slope       0
     ca          0
     thal        0
     target      0
     dtype: int64
```

10. df.columns

```
[8]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
            'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
           dtype='object')
```

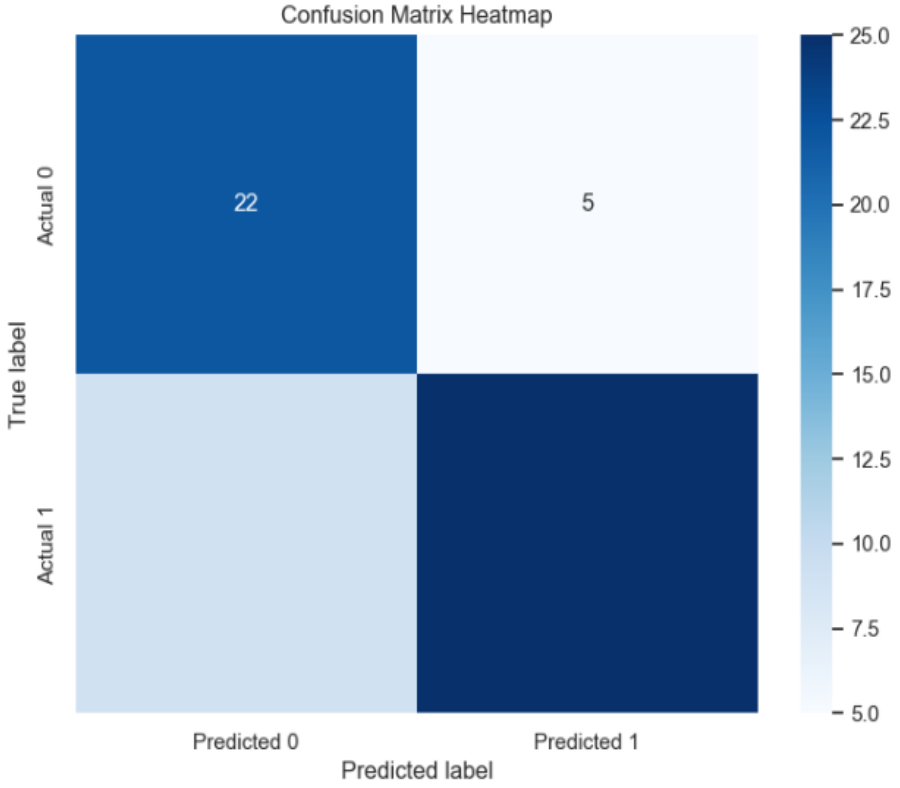
11. plt.figure(figsize=(10,10))
12. sea.pairplot(df,height=1.5,hue='target')
13. plt.show()



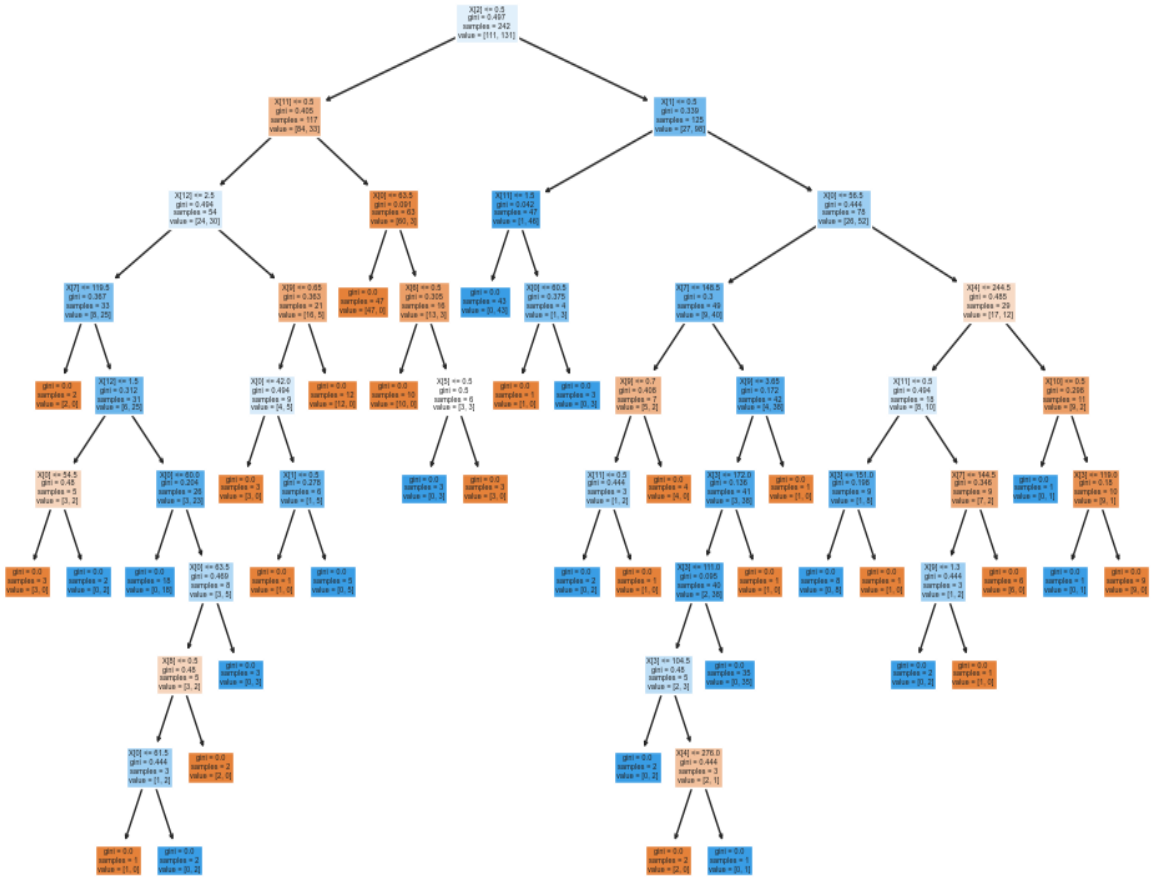
14. X=df.iloc[:,0:-1] ## independent features
15. y=df.iloc[:,-1] ## dependent features
16. from sklearn.model_selection import train_test_split
17. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
18. from sklearn.tree import DecisionTreeClassifier
19. DT=DecisionTreeClassifier()
20. DT_model=DT.fit(X_train,y_train)
21. y_pred = DT.predict(X_test)
22. from sklearn.metrics import r2_score,confusion_matrix,accuracy_score
23. cm=confusion_matrix(y_test,y_pred)
24. print(cm)
25. accuracy_score(y_test, y_pred)

```
[[22  5]
 [ 9 25]]
[47]: 0.7704918032786885
```

26. plt.figure(figsize=(8, 6))
27. sea.heatmap(cm, annot=True, cmap='Blues', fmt='g',
28. xticklabels=['Predicted 0', 'Predicted 1'],
29. yticklabels=['Actual 0', 'Actual 1'])
30. plt.xlabel('Predicted label')
31. plt.ylabel('True label')
32. plt.title('Confusion Matrix Heatmap')
33. plt.show()



34. from sklearn.tree import plot_tree
35. plt.figure(figsize=(12, 8))
36. plot_tree(DT, filled=True)
37. plt.show()



Learning Outcome: This project deepens understanding of decision tree classification an d its application in diagnosing heart disease. Participants acquire skills in data preprocess ing and model evaluation, interpreting insights from an accuracy score of 77%. Insights in form healthcare interventions, facilitating early detection and targeted car to improve patient outcomes and reduce heart-related morbidity and mortality.

Experiment 5: Random Forest Classifier

Abstract: This study investigates the application of random forest classification on heart disease data to predict the presence or absence of heart disease. Through rigorous data preprocessing and feature engineering, random forest models were trained and evaluated, resulting in an impressive accuracy score of 86.8%. These findings highlight the effectiveness of random forest algorithms in diagnosing heart disease. The study's implications extend to improving healthcare outcomes through early detection and intervention strategies. By leveraging random forest analysis, healthcare practitioners can identify individuals at risk of heart disease and implement targeted interventions to mitigate adverse health outcomes. Overall, this study underscores the importance of data-driven approaches in enhancing cardiac care and reducing the burden of heart-related morbidity and mortality.

Code and Output

1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import seaborn as sea
5. df=pd.read_csv(r'C:\Users\Dell\Desktop\FML\heart_desease_data.csv')
6. df.head()

[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

7. df.describe()

[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.54455
	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.49883
	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.00000
	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.00000
	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.00000
	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.00000

8. df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

9. df.isnull().sum()

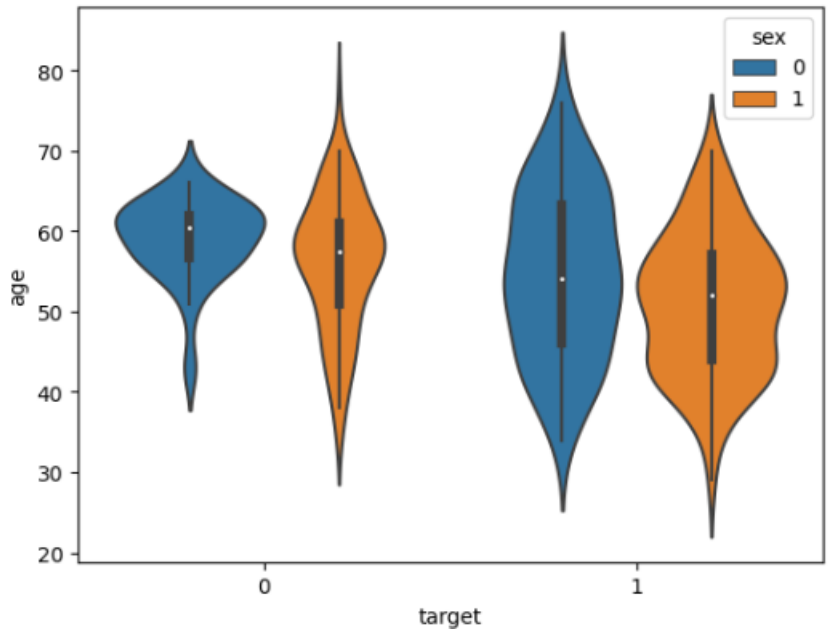
```
[6]: age         0
     sex         0
     cp          0
     trestbps    0
     chol        0
     fbs         0
     restecg     0
     thalach     0
     exang       0
     oldpeak     0
     slope       0
     ca          0
     thal        0
     target      0
     dtype: int64
```

10.
11. df.columns

```
[8]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
            'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
           dtype='object')
```

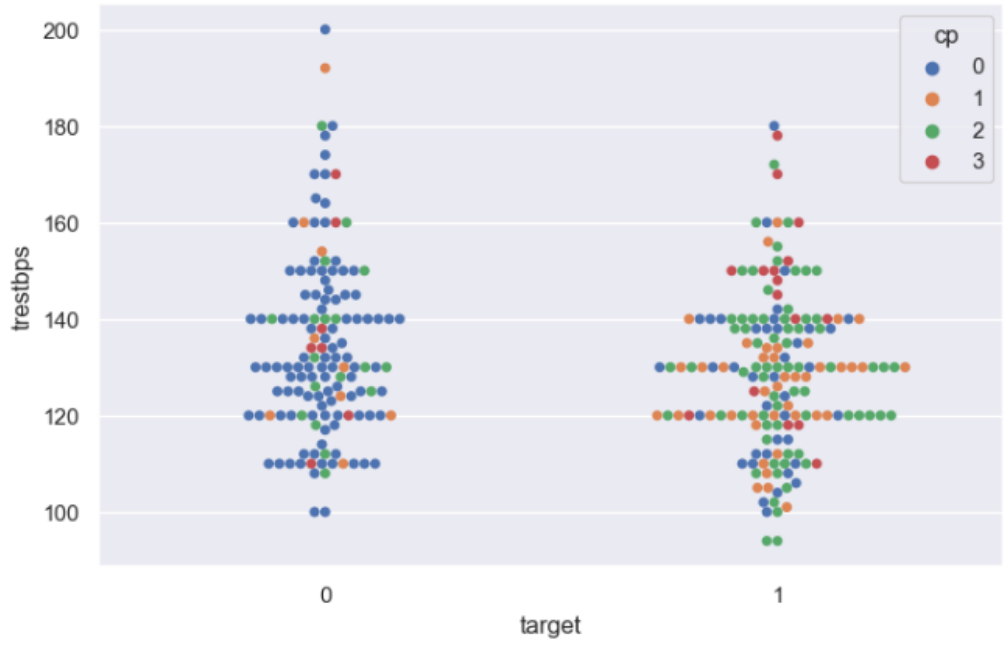
12.
13. sea.violinplot(data=df, x="target", y="age", hue="sex")

```
[13]: <Axes: xlabel='target', ylabel='age'>
```



14. sea.swarmplot(data=df, x="target", y="trestbps", hue="cp", palette="deep")

```
[38]: <Axes: xlabel='target', ylabel='trestbps'>
```

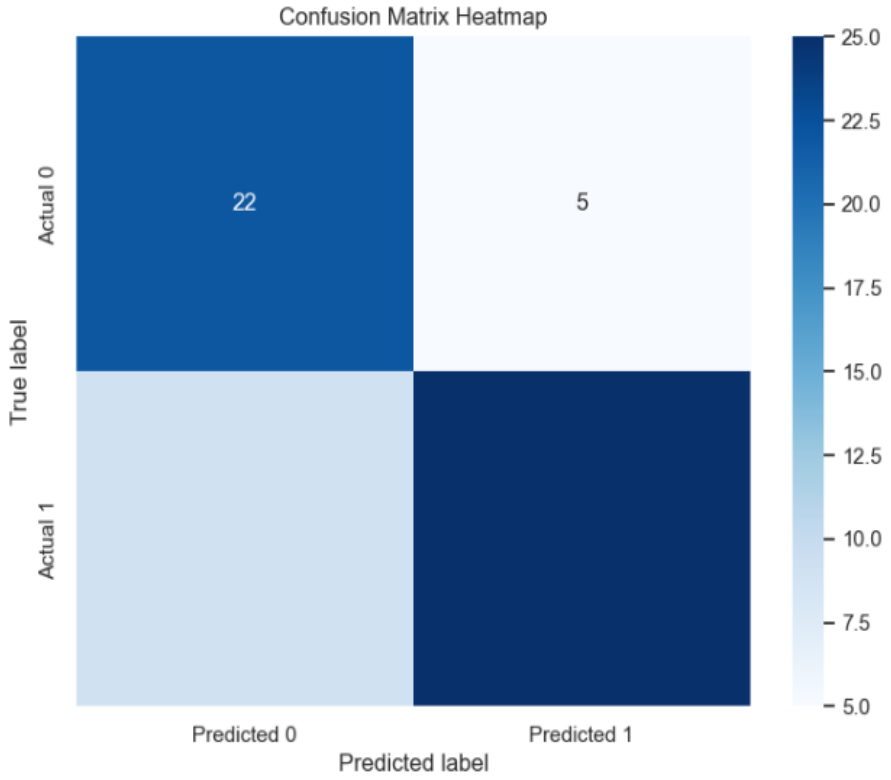


15. X=df.iloc[:,0:-1] ## independent features
16. y=df.iloc[:,-1] ## dependent features
17. from sklearn.model_selection import train_test_split
18. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
19. from sklearn.ensemble import RandomForestClassifier

20. RF=RandomForestClassifier()
21. RF.fit(X_train, y_train) y_pred = RF.predict(X_test)
22. cm=confusion_matrix(y_test,y_pred)
23. print(cm)
24. accuracy_score(y_test, y_pred)

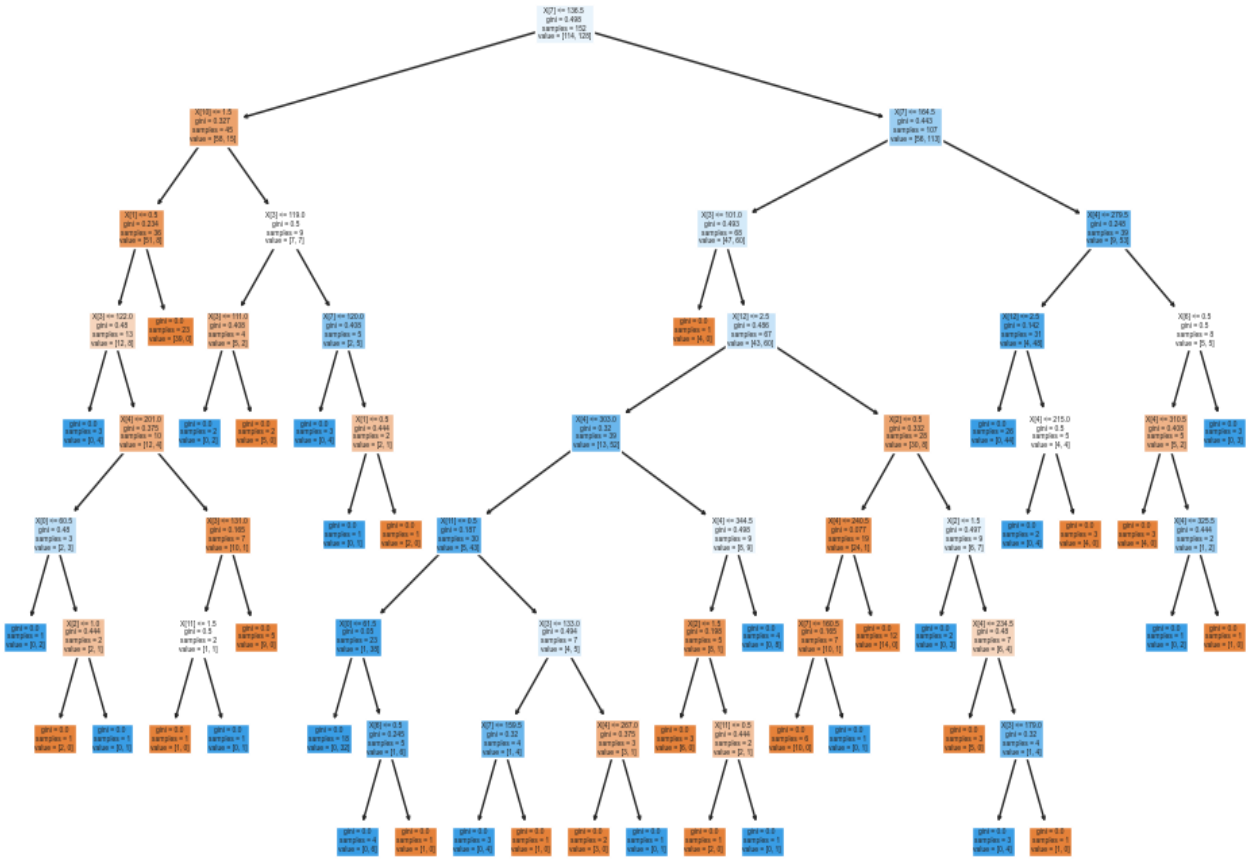
```
[[22  5]
 [ 3 31]]
[41]:  0.8688524590163934
```

25. plt.figure(figsize=(8, 6))
26. sea.heatmap(cm, annot=True, cmap='Blues', fmt='g',
27. xticklabels=['Predicted 0', 'Predicted 1'],
28. yticklabels=['Actual 0', 'Actual 1'])
29. plt.xlabel('Predicted label')
30. plt.ylabel('True label')
31. plt.title('Confusion Matrix Heatmap')
32. plt.show()



33. from sklearn.tree import plot_tree
34. plt.figure(figsize=(12, 8))
35. plot_tree(RF.estimators_[0],, filled=True)

36. plt.show()



Learning Outcome: This project enhances understanding of random forest classification a nd its application in diagnosing heart disease. Participants develop proficiency in data preprocessing and model evaluation, interpreting insights from an accuracy score of 86.8%. Insights inform healthcare interventions, facilitating early detection and targeted care to improve patient outcomes and reduce heart-related morbidity and mortality.

Experiment 6: Support Vector Classifier

Abstract: This study investigates the application of support vector classification (SVC) on a thyroid dataset to classify thyroid-related conditions. The dataset comprises various physiological and biochemical features relevant to thyroid health, including hormone levels, age, and gender. Through meticulous data preprocessing, missing values handling, and feature selection, SVC models were trained and evaluated. The study achieved an impressive accuracy score of 89.47%, indicating the effectiveness of SVC in diagnosing thyroid disorders. The findings of this study have significant implications for healthcare practice. Accurate diagnosis of thyroid conditions is crucial for effective management and treatment. By leveraging SVC analysis, healthcare practitioners can enhance diagnostic accuracy and provide timely interventions for individuals with thyroid disorders. Early detection of thyroid abnormalities enables proactive measures to prevent complications and improve patient outcomes. Furthermore, the study highlights the importance of machine learning techniques in medical diagnostics. SVC algorithms offer robust classification capabilities, particularly in complex datasets with nonlinear relationships. The success of SVC in classifying thyroid-related conditions underscores its potential as a valuable tool in healthcare decision-making.

Code and Output

1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import seaborn as sea
5. df=pd.read_csv(r'C:\Users\Dell\Desktop\FML\Thyroid_train.csv')
6. df.head()

[54]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

7. df.describe()

[55]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.627417
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.483918
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.000000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	1.000000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	1.000000

8. df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   mean_radius      569 non-null    float64
 1   mean_texture     569 non-null    float64
 2   mean_perimeter   569 non-null    float64
 3   mean_area        569 non-null    float64
 4   mean_smoothness  569 non-null    float64
 5   diagnosis        569 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

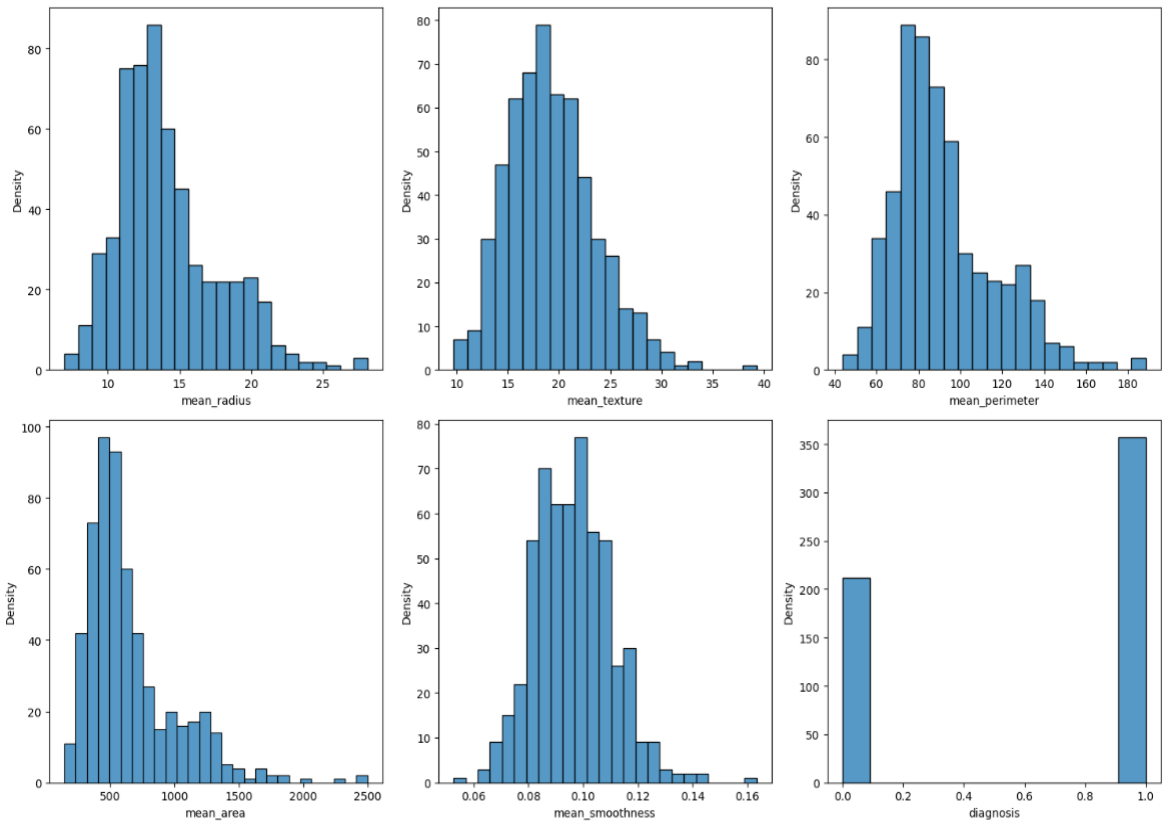
9. df.isnull().sum()

```
[57]: mean_radius        0
      mean_texture       0
      mean_perimeter     0
      mean_area          0
      mean_smoothness    0
      diagnosis          0
      dtype: int64
```

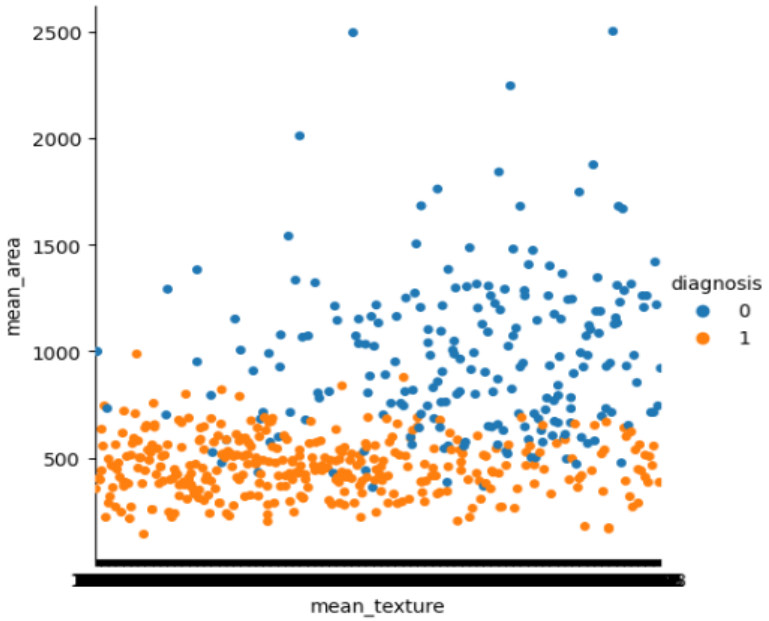
10. df.columns

```
[59]: Index(['mean_radius', 'mean_texture', 'mean_perimeter', 'mean_area',
             'mean_smoothness', 'diagnosis'],
            dtype='object')
```

11. fig, axes = plt.subplots(2, 3, figsize=(15, 10))
12. axes = axes.flatten()
13. for i, column in enumerate(df.columns):
14. sea.histplot(df[column], ax=axes[i])
15. axes[i].set_xlabel(column)
16. axes[i].set_ylabel('Density')
17. plt.tight_layout()
18. plt.show()



19. sea.catplot(data=df, x="mean_texture", y="mean_area", hue="diagnosis")

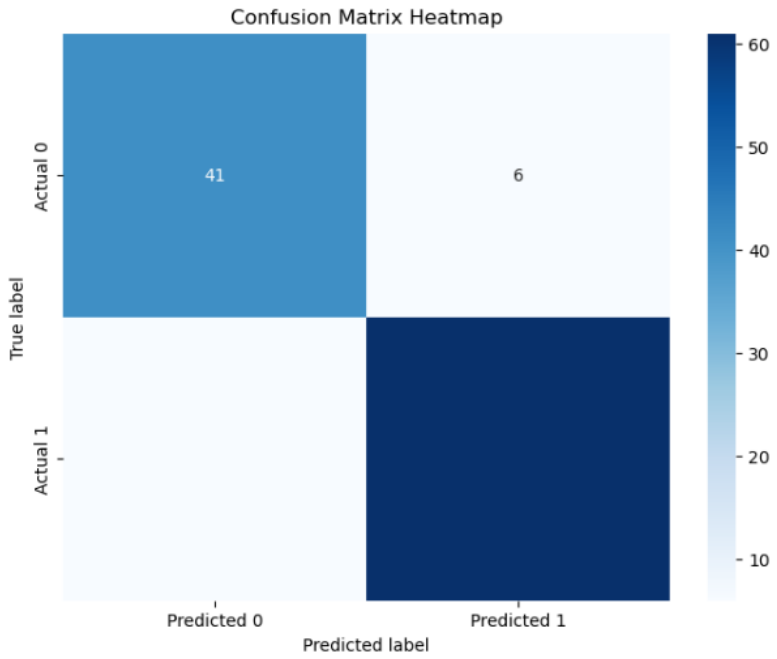


20. X=df.iloc[:,0:-1] ## independent features
21. y=df.iloc[:,-1] ## dependent features
22. from sklearn.model_selection import train_test_split

```
23. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
24. from sklearn.preprocessing import StandardScaler
25. sc=StandardScaler()
26. X_train=sc.fit_transform(X_train)
27. X_test=sc.transform(X_test)
28. from sklearn.svm import SVC
29. classifier = SVC(kernel = 'rbf', random_state = 0)
30. SVC_model=classifier.fit(X_train, y_train)
31. from sklearn.metrics import r2_score,confusion_matrix,accuracy_score
32. y_pred = classifier.predict(X_test)
33. cm=confusion_matrix(y_test,y_pred)
34. print(cm)
35. accuracy_score(y_test, y_pred)
```

```
[[41  6]
 [ 6 61]]
[65]:  0.8947368421052632
```

```
36. plt.figure(figsize=(8, 6))
37. sea.heatmap(cm, annot=True, cmap='Blues', fmt='g',
38. xticklabels=['Predicted 0', 'Predicted 1'],
39. yticklabels=['Actual 0', 'Actual 1'])
40. plt.xlabel('Predicted label')
41. plt.ylabel('True label')
42. plt.title('Confusion Matrix Heatmap')
43. plt.show()
```



Learning Outcome:

This project enhances understanding of support vector classification (SVC) in thyroid diagnosis. Participants interpret an accuracy score of 89.47%, gaining insights into SVC's efficacy. Results inform personalized treatment plans, improving patient outcomes in thyroid disease management. Participants develop critical thinking skills and readiness to apply SVC algorithms in healthcare practice, contributing to advancements in medical diagnostics and enhancing patient care.

Experiment 7: K-Means Clustering

Abstract: This study explores the application of K-means clustering on breast cancer data to identify distinct clusters of symptoms associated with the disease. Using a dataset comprising various clinical and pathological features, K-means clustering was employed to group patients into four clusters based on symptom similarity. The analysis revealed distinct symptom profiles within each cluster, providing valuable insights into the heterogeneity of breast cancer manifestations. These findings have significant implications for personalized treatment approaches and patient care management. By leveraging K-means clustering, healthcare practitioners can better understand the diverse symptomatology of breast cancer and tailor treatment strategies accordingly. Overall, this study underscores the importance of data-driven clustering techniques in oncology research and highlights the potential of K-means clustering in facilitating precision medicine for breast cancer patients.

Code and Output

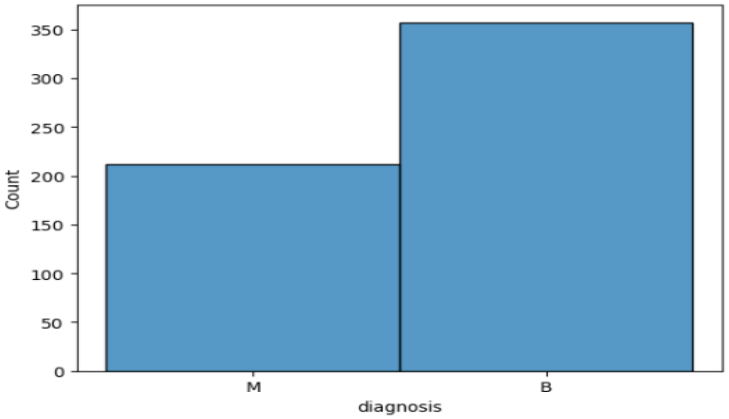
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. import seaborn as sea
5. df = pd.read_csv(r'C:\Users\Dell\Desktop\FML\data.csv')
6. df.head()

[28]:

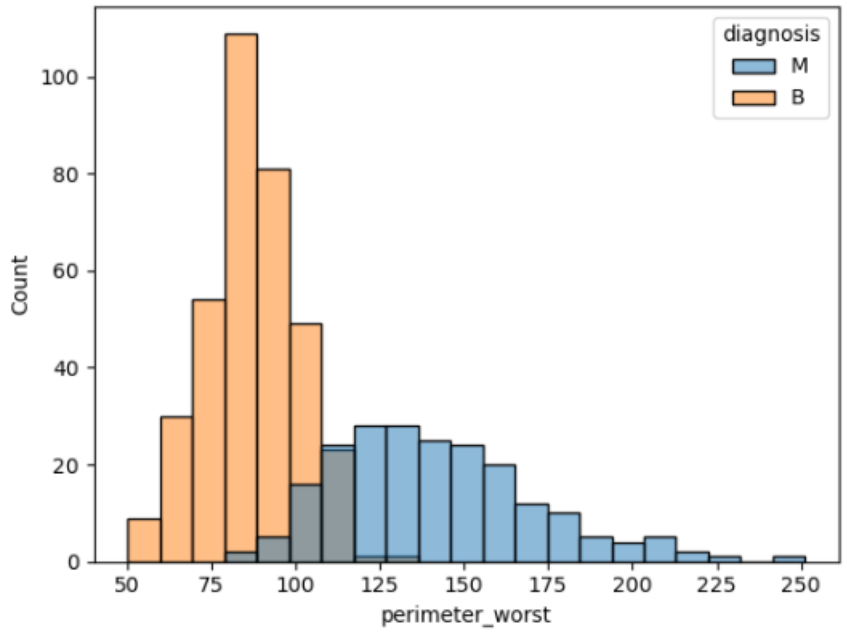
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

5 rows × 32 columns

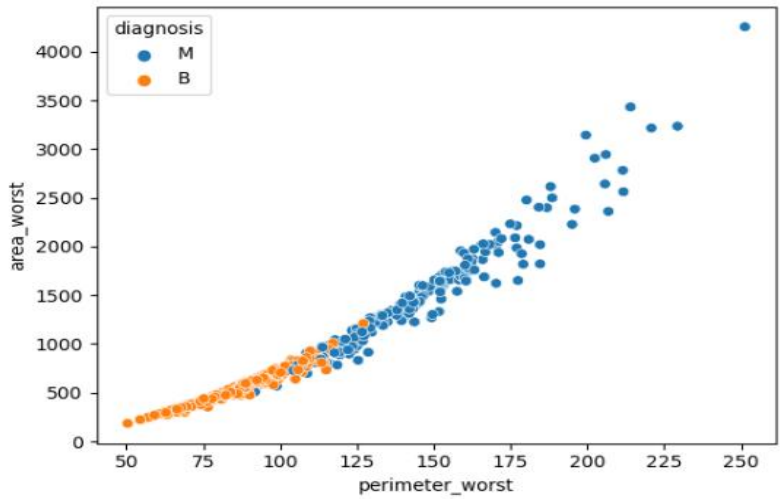
7. sea.histplot(df['diagnosis']);



8. sea.histplot(data=df, x='perimeter_worst', hue='diagnosis');

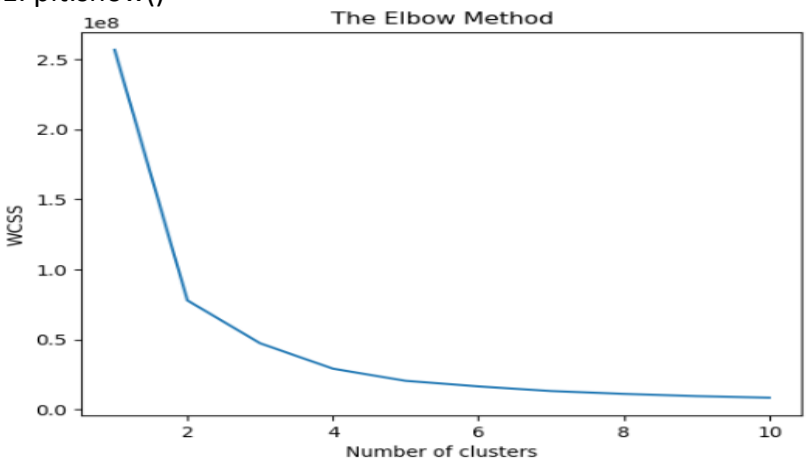


9. sea.scatterplot(x=df['perimeter_worst'],y=df['area_worst'], hue=df['diagnosis']);

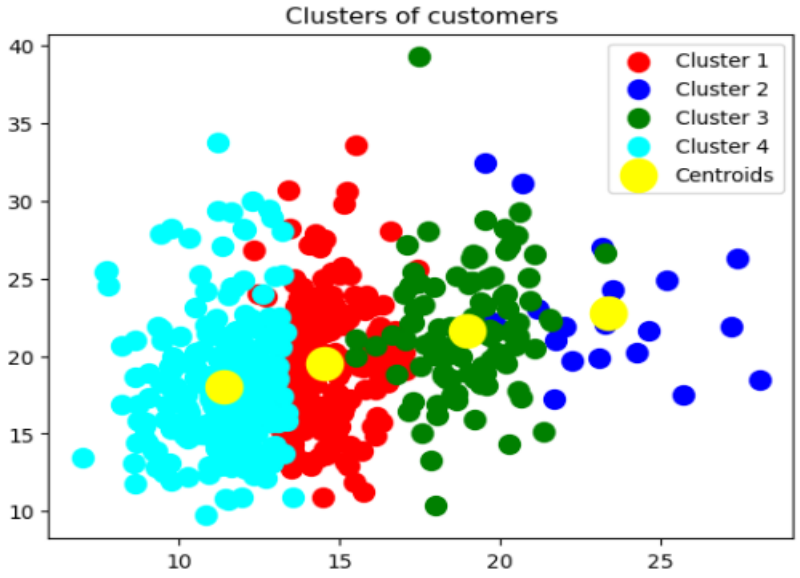


10. X = df.iloc[:, 2:-1].values
11. from sklearn.cluster import KMeans
12. wcss = []
13. for i in range(1, 11):
14. kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
15. kmeans.fit(X)
16. wcss.append(kmeans.inertia_)
17. plt.plot(range(1, 11), wcss)
18. plt.title('The Elbow Method')
19. plt.xlabel('Number of clusters')
20. plt.ylabel('WCSS')

21. plt.show()



22. kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
23. y_kmeans = kmeans.fit_predict(X)
24. plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluste1')
25. plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
26. plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
27. plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
28. plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
29. plt.title('Clusters of Symptoms)
30. plt.legend()
31. plt.show()



Learning Outcome: Participants gain understanding of K-means clustering in breast cancer analysis, interpreting distinct symptom clusters. They develop critical thinking skills, recognizing heterogeneity in cancer manifestations. Insights inform personalized treatment approaches, facilitating precision medicine in oncology care.