AI and ML TASK 4

Importing libraries

In [59]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
dat=pd.read_csv('data.csv')
```

In [5]:
```
data
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	17.33	184.60	2019.0	0.16220	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	23.41	158.80	1956.0	0.12380	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	25.53	152.50	1709.0	0.14440	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	26.50	98.87	567.7	0.20980	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	16.67	152.20	1575.0	0.13740	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	26.40	166.10	2027.0	0.14100	0
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	38.25	155.00	1731.0	0.11660	0
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	34.12	126.70	1124.0	0.11390	0
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	39.42	184.60	1821.0	0.16500	0
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	30.37	59.16	268.6	0.08996	0

569 rows × 33 columns

In [9]:
```
data.describe()
```

Out[9]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_wor
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	25.677223	107.261213	880.583128	0.13236
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	6.146258	33.602542	569.356993	0.02282
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	12.020000	50.410000	185.200000	0.07117
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	21.080000	84.110000	515.300000	0.11660
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	25.410000	97.660000	686.500000	0.13130
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	29.720000	125.400000	1084.000000	0.14600
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	49.540000	251.200000	4254.000000	0.22260

8 rows × 32 columns

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

checking for missing values

In [13]:
```
missing = data.isnull().sum()
```

In [15]:
```
missing
```

Out[15]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

handling categorical data

In [17]:
```
data_cleaned = data.drop(columns=['id', "Unnamed: 32"])
```

In [19]:
```
data_cleaned["diagnosis"] = data_cleaned["diagnosis"].map({"M": 1, "B": 0})
```

```
data_cleaned.head()
```

Train/test split and standardize features.

In [25]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X = data_cleaned.drop(columns=["diagnosis"])
y = data_cleaned["diagnosis"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [27]:
```
X_train_scaled.shape, X_test_scaled.shape, y_train.shape, y_test.shape
```

Out[27]:
```
((455, 30), (114, 30), (455,), (114,))
```

In [33]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
print(f"Accuracy:", accuracy_score(y_test, y_pred))
print(f"\nClassification Report:\n", classification_report(y_test, y_pred))
print(f"\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9736842105263158

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114


Confusion Matrix:
 [[70  1]
 [ 2 41]]
```

Fit a Logistic Regression model.

In [35]:
```
from sklearn.metrics import (
    confusion_matrix, classification_report, roc_auc_score, roc_curve, accuracy_score
)
```

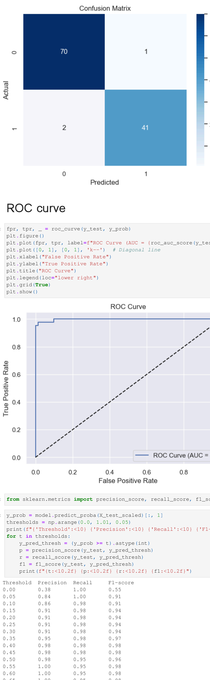
In [37]:
```
model = LogisticRegression(random_state=42, max_iter=1000) # for traing logistic regression
model.fit(X_train_scaled, y_train)
```

Out[37]:
```
LogisticRegression(max_iter=1000, random_state=42)
```

In [41]:
```
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:, 1]  # Probabilities for the positive class
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob))
```

```
Accuracy: 0.9736842105263158

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

ROC-AUC Score: 0.9973798245598679
```

In [43]:
```
cm = confusion_matrix(y_test, y_pred) # for confusion matrix
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



ROC curve

In [45]:
```
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure()
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc_score(y_test, y_prob):.2f})")
plt.plot([0, 1], [0, 1], "k--")  # Diagonal line
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```


In [57]:
```
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [61]:
```
y_prob = model.predict_proba(X_test_scaled)[:, 1]
thresholds = np.arange(0.0, 1.01, 0.05)
print(f"{'Threshold':<10} {'Precision':<10} {'Recall':<10} {'F1-score':<10}")
for t in thresholds:
    y_pred_thresh = (y_prob >= t).astype(int)
    p = precision_score(y_test, y_pred_thresh)
    r = recall_score(y_test, y_pred_thresh)
    f1 = f1_score(y_test, y_pred_thresh)
    print(f"{t:<10.2f} {p:<10.2f} {r:<10.2f} {f1:<10.2f}")
```

```
Threshold  Precision  Recall     F1-score
0.00       0.38       1.00       0.55
0.05       0.84       1.00       0.91
0.10       0.96       0.98       0.97
0.15       0.95       0.98       0.96
0.20       0.95       0.98       0.96
0.25       0.95       0.98       0.96
0.30       0.95       0.98       0.96
0.35       0.95       0.98       0.97
0.40       0.98       0.98       0.98
0.45       0.98       0.98       0.98
0.50       0.98       0.95       0.96
0.55       1.00       0.95       0.98
0.60       1.00       0.95       0.98
0.65       1.00       0.95       0.98
0.70       1.00       0.95       0.98
0.75       1.00       0.93       0.96
0.80       1.00       0.93       0.96
0.85       1.00       0.91       0.95
0.90       1.00       0.88       0.94
```

```
0.95      1.00      0.86      0.93
1.00      0.00      0.00      0.00
```

In [ ]: