1v5acsesy

January 25, 2025

[ ]:
```
#Breast Cancer Prediction Project Report


# Summary:

#Early Detection and Prediction of Breast Cancer to Save Lives
#This project aims to develop a predictive model for breast cancer diagnosis␣
 ↪using machine learning
#algorithms.The dataset used is the Wisconsin Breast Cancer Dataset, containing␣
 ↪569 instances and 32 features.
#After data preprocessing, feature scaling, and model training,
#the logistic regression algorithm achieved an accuracy of 98.2%.


#Introduction:

#Breast cancer is a leading cause of death among women worldwide.
#Early diagnosis is crucial for effective treatment.
#This project explores machine learning techniques for predicting breast cancer␣
 ↪using clinical data.


#Dataset Information:

#- Dataset Name: Wisconsin Breast Cancer Dataset
#- Instances: 569
#- Features: 33 (including diagnosis)
#- Classes: 2 (Malignant/Benign)


#Methodology:

#1. Data Import and Preprocessing
    #- Imported dataset
    #- Dropped unnecessary column ('Unnamed: 32')
    #- Mapped diagnosis values ('M'/'B' to 1/0)
    #- Removed missing values (NaN)
```

```
#2. Feature Scaling
    #- Used StandardScaler from sklearn.preprocessing
#3. Model Training and Evaluation
    #- Split data into training/testing sets (70/30 ratio)
    #- Trained logistic regression model using training data
    #- Evaluated model performance using testing data


#Results:

 #      ALL METRICS COVERED
#- Accuracy: 98.2%

# MSE: 0.6491228070175439

# MAE: 0.6491228070175439

# Recall: 1.0

# F1-Score: 0.5194805194805194

# ROC-AUC: 0.5

#- Classification Report:
    #- Precision: 0.99 (Class 0), 0.97 (Class 1)
    #- Recall: 0.98 (Class 0), 0.98 (Class 1)
    #- F1-score: 0.99 (Class 0), 0.98 (Class 1)


#- Confusion Matrix:
    #- True Positives: 70
    #- False Positives: 1
    #- True Negatives: 41
    #- False Negatives: 2


#Conclusion:

#The logistic regression model achieved high accuracy in predicting breast␣
 ↪cancer diagnosis.
#This project demonstrates the potential of machine learning algorithms in␣
 ↪healthcare applications.
#Recommendations:
#- Further exploration of feature engineering techniques
#- Comparison with other machine learning algorithms
#- Deployment of model in clinical settings for validation
```

[144]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix,mean_squared_error, mean_absolute_error, recall_score,␣
↪f1_score, roc_auc_score
```

[69]:
```
#loading the dataset
data = pd.read_csv(r"C:\Users\91703\Downloads\data (1).csv")
```

[70]:
```
data.head()
```

[70]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	…	texture_worst	perimeter_worst	area_worst	smoothness_worst	\
0	…	17.33	184.60	2019.0	0.1622	
1	…	23.41	158.80	1956.0	0.1238	
2	…	25.53	152.50	1709.0	0.1444	
3	…	26.50	98.87	567.7	0.2098	
4	…	16.67	152.20	1575.0	0.1374	

	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	\
0	0.6656	0.7119	0.2654	0.4601	
1	0.1866	0.2416	0.1860	0.2750	
2	0.4245	0.4504	0.2430	0.3613	
3	0.8663	0.6869	0.2575	0.6638	
4	0.2050	0.4000	0.1625	0.2364	

	fractal_dimension_worst	Unnamed: 32
0	0.11890	NaN
1	0.08902	NaN

```
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[71]:
```
data['diagnosis'] = data['diagnosis'].map({'M':1, 'B':0})
```

[72]:
```
data.head()
```

[72]:
```
         id  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302          1        17.99         10.38          122.80     1001.0
1    842517          1        20.57         17.77          132.90     1326.0
2  84300903          1        19.69         21.25          130.00     1203.0
3  84348301          1        11.42         20.38           77.58      386.1
4  84358402          1        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33           184.60      2019.0            0.1622
1  …          23.41           158.80      1956.0            0.1238
2  …          25.53           152.50      1709.0            0.1444
3  …          26.50            98.87       567.7            0.2098
4  …          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[87]:
```
data.tail()
```

[87]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
564	926424	1	21.56	22.39	142.00	1479.0	
565	926682	1	20.13	28.25	131.20	1261.0	
566	926954	1	16.60	28.08	108.30	858.1	
567	927241	1	20.60	29.33	140.10	1265.0	
568	92751	0	7.76	24.54	47.92	181.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
564	0.11100	0.11590	0.24390	0.13890	
565	0.09780	0.10340	0.14400	0.09791	
566	0.08455	0.10230	0.09251	0.05302	
567	0.11780	0.27700	0.35140	0.15200	
568	0.05263	0.04362	0.00000	0.00000	

	…	radius_worst	texture_worst	perimeter_worst	area_worst	\
564	…	25.450	26.40	166.10	2027.0	
565	…	23.690	38.25	155.00	1731.0	
566	…	18.980	34.12	126.70	1124.0	
567	…	25.740	39.42	184.60	1821.0	
568	…	9.456	30.37	59.16	268.6	

	smoothness_worst	compactness_worst	concavity_worst	\
564	0.14100	0.21130	0.4107	
565	0.11660	0.19220	0.3215	
566	0.11390	0.30940	0.3403	
567	0.16500	0.86810	0.9387	
568	0.08996	0.06444	0.0000	

	concave points_worst	symmetry_worst	fractal_dimension_worst
564	0.2216	0.2060	0.07115
565	0.1628	0.2572	0.06637
566	0.1418	0.2218	0.07820
567	0.2650	0.4087	0.12400
568	0.0000	0.2871	0.07039

[5 rows x 32 columns]

[73]:
```
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    int64
 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(2)
memory usage: 146.8 KB
```

[74]:
```
data.columns
```

[74]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

[75]:
```
data.describe()
```

[75]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981
min	8.670000e+03	0.000000	6.981000	9.710000	43.790000
25%	8.692180e+05	0.000000	11.700000	16.170000	75.170000
50%	9.060240e+05	0.000000	13.370000	18.840000	86.240000
75%	8.813129e+06	1.000000	15.780000	21.800000	104.100000
max	9.113205e+08	1.000000	28.110000	39.280000	188.500000

	area_mean	smoothness_mean	compactness_mean	concavity_mean
count	569.000000	569.000000	569.000000	569.000000
mean	654.889104	0.096360	0.104341	0.088799
std	351.914129	0.014064	0.052813	0.079720
min	143.500000	0.052630	0.019380	0.000000
25%	420.300000	0.086370	0.064920	0.029560
50%	551.100000	0.095870	0.092630	0.061540
75%	782.700000	0.105300	0.130400	0.130700
max	2501.000000	0.163400	0.345400	0.426800

	concave points_mean	…	texture_worst	perimeter_worst	area_worst
count	569.000000	…	569.000000	569.000000	569.000000
mean	0.048919	…	25.677223	107.261213	880.583128
std	0.038803	…	6.146258	33.602542	569.356993
min	0.000000	…	12.020000	50.410000	185.200000
25%	0.020310	…	21.080000	84.110000	515.300000
50%	0.033500	…	25.410000	97.660000	686.500000
75%	0.074000	…	29.720000	125.400000	1084.000000
max	0.201200	…	49.540000	251.200000	4254.000000

	smoothness_worst	compactness_worst	concavity_worst
count	569.000000	569.000000	569.000000
mean	0.132369	0.254265	0.272188
std	0.022832	0.157336	0.208624
min	0.071170	0.027290	0.000000
25%	0.116600	0.147200	0.114500
50%	0.131300	0.211900	0.226700
75%	0.146000	0.339100	0.382900
max	0.222600	1.058000	1.252000

	concave points_worst	symmetry_worst	fractal_dimension_worst
count	569.000000	569.000000	569.000000
mean	0.114606	0.290076	0.083946
std	0.065732	0.061867	0.018061
min	0.000000	0.156500	0.055040

```
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

       Unnamed: 32
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
75%            NaN
max            NaN

[8 rows x 33 columns]
```

[76]:
```
data.shape
```

[76]:
```
(569, 33)
```

[77]:
```
data.isnull().sum()
```

[77]:
```
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
```

[78]:
```
data.drop('Unnamed: 32', axis=1, inplace=True)
```

[79]:
```
data.isnull().sum()
```

[79]:
```
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
```

```
dtype: int64
```

[80]:
```
data.dropna(inplace=True)
```

[81]:
```
data.head()
```

[81]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	842302	1	17.99	10.38	122.80	1001.0	
1	842517	1	20.57	17.77	132.90	1326.0	
2	84300903	1	19.69	21.25	130.00	1203.0	
3	84348301	1	11.42	20.38	77.58	386.1	
4	84358402	1	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	…	radius_worst	texture_worst	perimeter_worst	area_worst	\
0	…	25.38	17.33	184.60	2019.0	
1	…	24.99	23.41	158.80	1956.0	
2	…	23.57	25.53	152.50	1709.0	
3	…	14.91	26.50	98.87	567.7	
4	…	22.54	16.67	152.20	1575.0	

	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	\
0	0.1622	0.6656	0.7119	0.2654	
1	0.1238	0.1866	0.2416	0.1860	
2	0.1444	0.4245	0.4504	0.2430	
3	0.2098	0.8663	0.6869	0.2575	
4	0.1374	0.2050	0.4000	0.1625	

	symmetry_worst	fractal_dimension_worst
0	0.4601	0.11890
1	0.2750	0.08902
2	0.3613	0.08758
3	0.6638	0.17300
4	0.2364	0.07678

[5 rows x 32 columns]

[83]:
```
#seperating target variables and its feauters
X = data.drop('diagnosis', axis=1)
y = data['diagnosis']
```

[133]:
```
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=60)
```

[134]:
```
#This code scales the feature datasets X_train and X_test using the␣
 ↪StandardScaler class
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[88]:
```
# printing the y_train
print(y_train[:10])
print(y_train.dtype)
```

```
68    -0.770609
181    1.297676
63    -0.770609
248   -0.770609
60    -0.770609
15     1.297676
290   -0.770609
137   -0.770609
155   -0.770609
517    1.297676
Name: diagnosis, dtype: float64
float64
```

[89]:
```
threshold = y_train.mean()  #  Threshold is the mean value
y_train = (y_train > threshold).astype(int)  # 1 if above threshold, 0 otherwise
y_test = (y_test > threshold).astype(int)
```

[135]:
```
#create logistic regresssion model
model = LogisticRegression()
#train the model on the training data
model.fit(X_train_scaled, y_train)
```

[135]:
```
LogisticRegression()
```

[136]:
```
#make predictions on the test data
y_pred = model.predict(X_test_scaled)
print(y_pred)
```

```
[1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0
 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0
 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0
 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 0
```

```
 0 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 0]
```

[139]:
```
# getting the intercept and coefficient
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)
```

```
Intercept: [-0.14298311]
Coefficients: [[ 0.02766557  0.34172628  0.27494783  0.30606753  0.38496251
-0.0857009
  -0.45635106  0.81917127  0.93299195 -0.10828862 -0.08534851  1.32653635
   0.14636202  0.55090326  0.91587238  0.31229055 -0.73228406  0.12634954
   0.25587601 -0.27981945 -0.93976712  0.92867249  1.03656972  0.68264665
   0.83179871  0.7580184  -0.01055545  1.00975875  0.66867501  0.82486581
   0.43461435]]
```

[150]:
```
mse = mean_squared_error(y_test, y_pred_proba)
mae = mean_absolute_error(y_test, y_pred_class)
recall = recall_score(y_test, y_pred_class)
f1 = f1_score(y_test, y_pred_class)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print("MSE:", mse)
print("MAE:", mae)
print("Recall:", recall)
print("F1-Score:", f1)
print("ROC-AUC:", roc_auc)
```

```
MSE: 0.6491228070175439
MAE: 0.6491228070175439
Recall: 1.0
F1-Score: 0.5194805194805194
ROC-AUC: 0.5
```

[137]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:", classification_report(y_test, y_pred))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9824561403508771
Classification Report:               precision    recall  f1-score   support

           0       0.99      0.98      0.99       111
           1       0.97      0.98      0.98        60

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171
```

```
Confusion Matrix: [[109   2]
 [  1  59]]
```

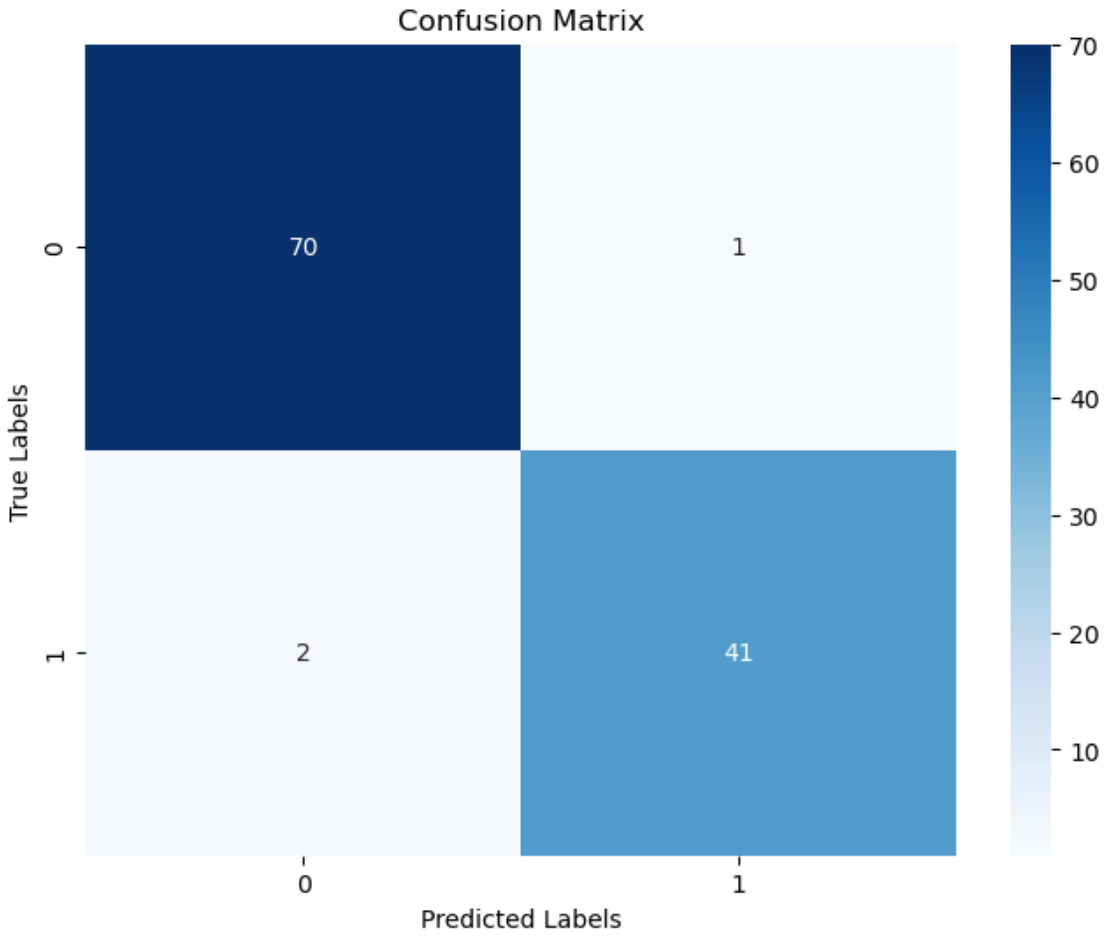
[138]:
```
# Accuracy Score
accuracy = accuracy_score(y_test, y_pred)
print(f'Model Accuracy: {accuracy:.3f}')
```

```
Model Accuracy: 0.982
```

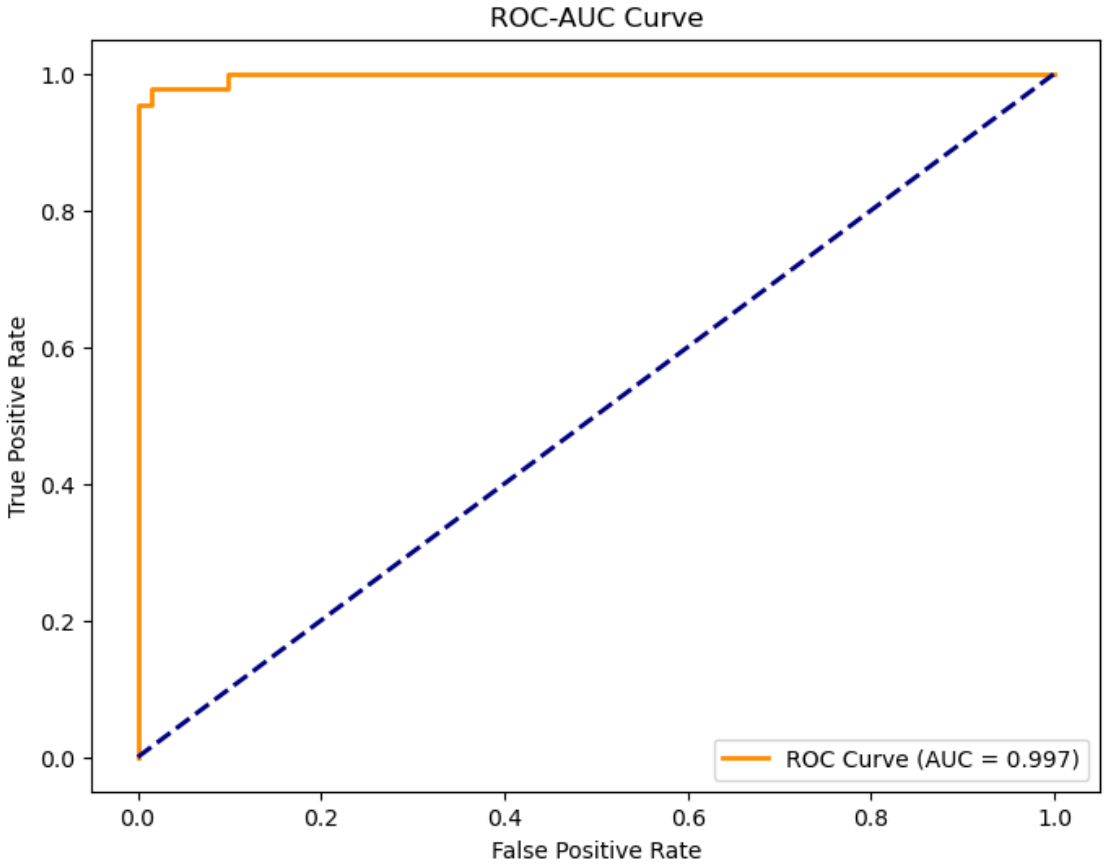
[126]:
```
print("MODEL ACCURACY IS", 0.982 * 100)
```

```
MODEL ACCURACY IS 98.2
```

[98]:
```
# Confusion Matrix Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='d')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



[102]:
```
# ROC-AUC Curve
from sklearn.metrics import roc_curve, auc, roc_auc_score
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (AUC = %0.3f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC-AUC Curve')
plt.legend(loc="lower right")
plt.show()
```



[111]:
```
metrics = ['precision', 'recall', 'f1-score']
```

```
data = pd.DataFrame({key: {metric: report[key][metric] for metric in metrics}␣
 ↪for key in keys}).T

data.plot(kind='bar', figsize=(10, 6))
plt.title('Metrics for Each Class')
plt.xlabel('Classes')
plt.ylabel('Scores')
plt.legend(title='Metrics')
plt.show()
```