```
# Importing necessary libraries for data manipulation and linear algebra
import pandas as pd  # For handling data
import numpy as np   # For numerical operations



# Settinging display options for easier data viewing
pd.set_option('display.max_columns', 100)


# Importing machine learning models and utilities from sklearn
from sklearn.model_selection import train_test_split, GridSearchCV, KFold, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc




# Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Setting a visual theme for seaborn for consistent visualizations
sns.set_style('whitegrid')



# loading data from a specified path
data_path = "/content/data (1).csv"
cancer_data = pd.read_csv(data_path)


# Brief exploration to understand the dataset's structure
print("Dataset dimensions:", cancer_data.shape)
print("First few rows of the dataset:")
print(cancer_data.head())
```

```
Dataset dimensions: (569, 33)
First few rows of the dataset:
         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

   symmetry_mean  fractal_dimension_mean  radius_se  texture_se  perimeter_se  \
0         0.2419                 0.07871     1.0950      0.9053         8.589
1         0.1812                 0.05667     0.5435      0.7339         3.398
2         0.2069                 0.05999     0.7456      0.7869         4.585
3         0.2597                 0.09744     0.4956      1.1560         3.445
4         0.1809                 0.05883     0.7572      0.7813         5.438

   area_se  smoothness_se  compactness_se  concavity_se  concave points_se  \
0   153.40       0.006399         0.04904       0.05373            0.01587
1    74.08       0.005225         0.01308       0.01860            0.01340
2    94.03       0.006150         0.04006       0.03832            0.02058
3    27.23       0.009110         0.07458       0.05661            0.01867
4    94.44       0.011490         0.02461       0.05688            0.01885

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0      0.03003              0.006193         25.38          17.33
1      0.01389              0.003532         24.99          23.41
2      0.02250              0.004571         23.57          25.53
3      0.05963              0.009208         14.91          26.50
```

```
4      0.01756              0.005115         22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656
1           158.80      1956.0            0.1238             0.1866
2           152.50      1709.0            0.1444             0.4245
3            98.87       567.7            0.2098             0.8663
4           152.20      1575.0            0.1374             0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
4           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
```

```
# Exploratory Data Analysis (EDA) - Checking for missing values
missing_values = cancer_data.isna().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
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
```

```
# Dropping columns with missing values
cancer_data_cleaned = cancer_data.dropna(axis=1)
print("Data after removing columns with missing values:", cancer_data_cleaned.shape)
```

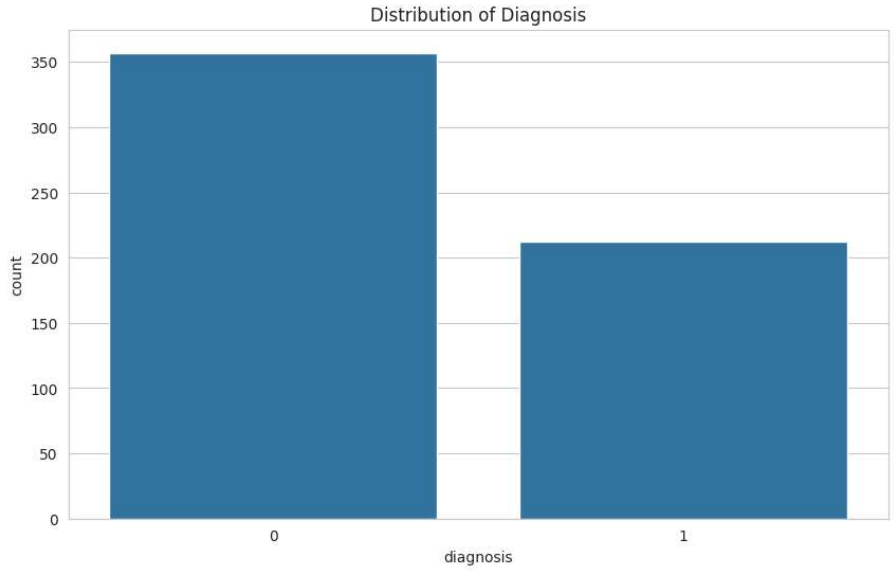
```
Data after removing columns with missing values: (569, 32)
```

```
 #Converting categorical data to numeric for analysis
le = LabelEncoder()
cancer_data_cleaned['diagnosis'] = le.fit_transform(cancer_data_cleaned['diagnosis'])
```

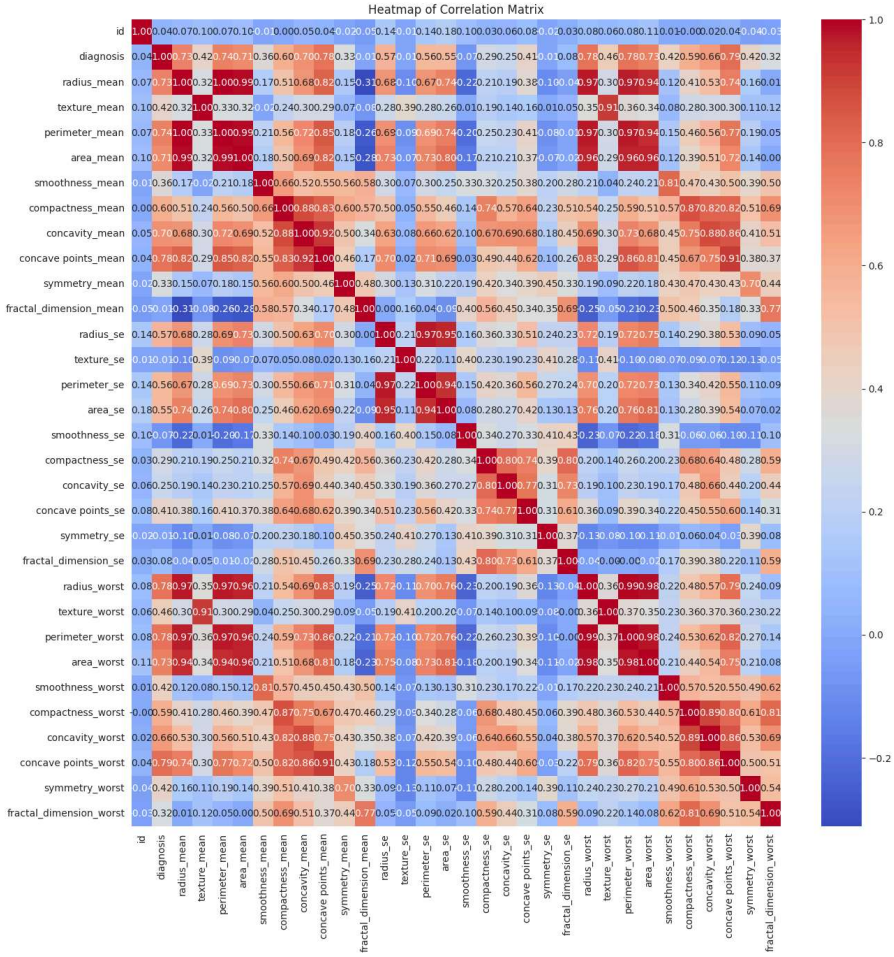
```
<ipython-input-13-c238d3200418>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
```

```
cancer_data_cleaned['diagnosis'] = le.fit_transform(cancer_data_cleaned['diagnosis'])
```

```
# Visualization of data to understand distribution
plt.figure(figsize=(10, 6))
sns.countplot(x='diagnosis', data=cancer_data_cleaned)
plt.title("Distribution of Diagnosis")
plt.show()
```

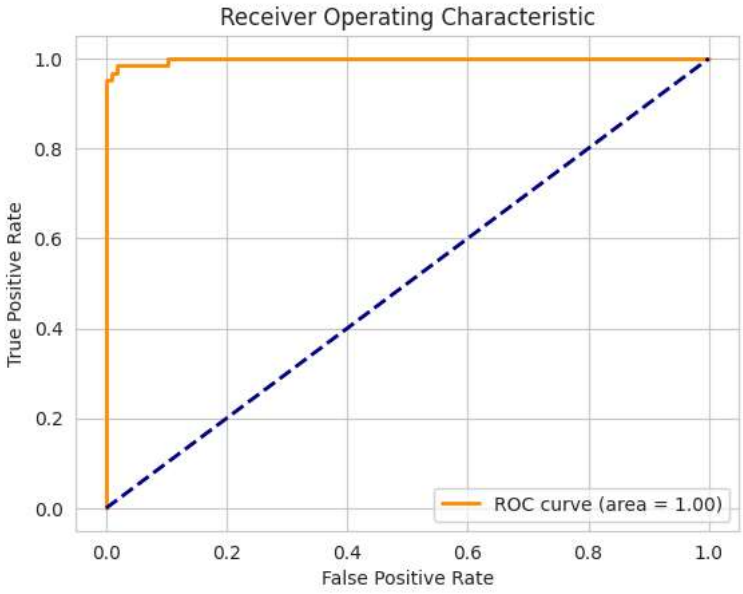



```
#  Heatmap of Correlation Matrix
corr_matrix = cancer_data_cleaned.corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Heatmap of Correlation Matrix")
plt.show()
```


Heatmap of Correlation Matrix
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
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

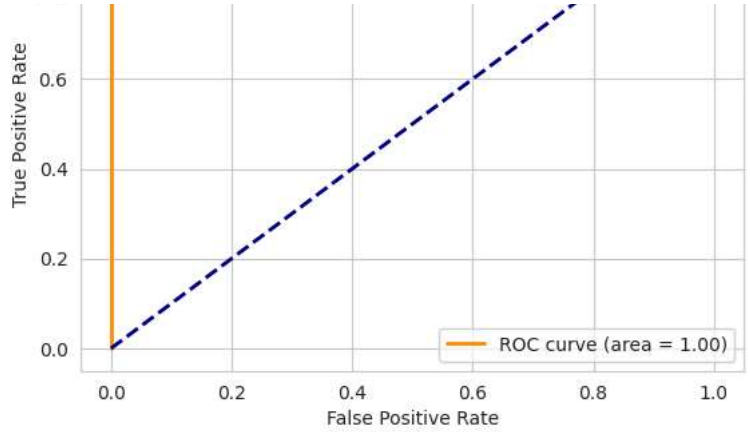
```
# Splitting data into training and testing sets
X = cancer_data_cleaned.drop('diagnosis', axis=1)
y = cancer_data_cleaned['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)



# Feature scaling for improved model performance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# Function to train and evaluate a model
def train_evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    print(f"Accuracy: {accuracy*100:.2f}%")
    print("Classification Report:")
    print(classification_report(y_test, predictions))
    # Additional Visualization: ROC Curve
    fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
    roc_auc = auc(fpr, tpr)
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % 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('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()



# Training and evaluating a Logistic Regression model
lr_model = LogisticRegression(max_iter=1000) # Increased max_iter for convergence
print("Logistic Regression Results:")
train_evaluate_model(lr_model, X_train_scaled, X_test_scaled, y_train, y_test)


# Additional models can be trained and evaluated using the same process
```

```
Logistic Regression Results:
Accuracy: 98.25%
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       108
           1       0.97      0.98      0.98        63

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

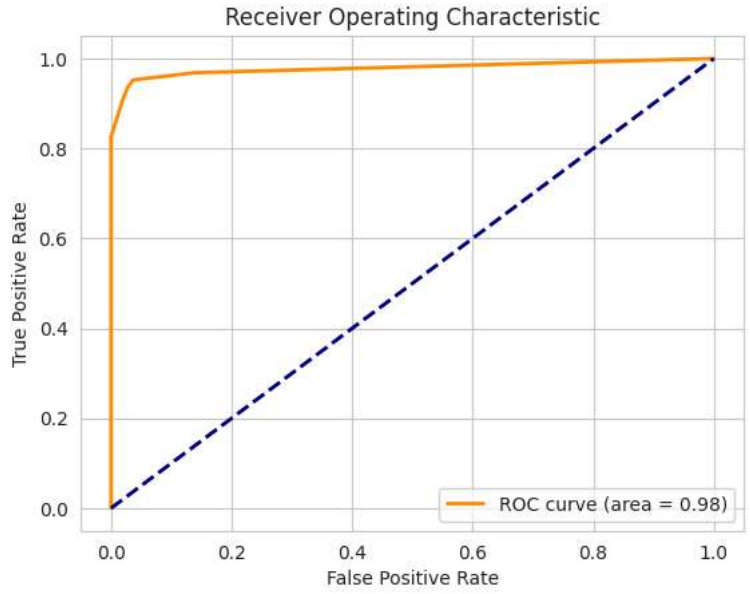



```
# using different models
models = {
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Machine': SVC(probability=True), # probability parameter set to True for ROC curve compatibility
    'K-Nearest Neighbors': KNeighborsClassifier(),
}
```

```
# Evaluating multiple models to find the best performer
for name, model in models.items():
    print(f"Evaluating model: {name}")
    train_evaluate_model(model, X_train_scaled, X_test_scaled, y_train, y_test)
```




```
Evaluating model: K-Nearest Neighbors
Accuracy: 95.91%
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       108
           1       0.95      0.94      0.94        63

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```



```
# Hyperparameter tuning for the best model (example with Random Forest)
param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
```