BREAST CANCER PREDICTION PROJECT

INTRODUCTION

Breast cancer is one of the most common cancers affecting women worldwide. Early and accurate diagnosis is crucial for effective treatment and improving patient outcomes. in this project i aim to develop a machine model to predict whether a breast tumor is a malignant or benign based on patient derived from digitized images of breast tissue. Using this dataset i will explore various features.

This project will follow **CRISP-DM** methodology:

1. **Business Understanding**
2. **Data Understanding**
3. **Data Preparation**
4. **Modeling**
5. **Evaluation**

Business Understanding

Problem statement

Breast cancer poses a significant health risk to women globally, with early detection being crucial for effective treatment and survival. Therefore i want to develop a reliable and accurate tool that can assist health care providers in identifying malignant(cancerous) tumors from benign(non-cancerous) ones using patient data

project objectives

1. Develop a predictive model
2. Improve diagnostic accuracy
3. Ensure the model's prediction are interpretable

Data Understanding

Dataset Name: Breast Cancer Wisconsin (Diagnostic) Data Set

Features

ID: Unique identifier

Diagnosis: The target variable

30 Numeric Features: predictors

In [ ]:

```
# import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [ ]:

```
# Loading the dataset
df = pd.read_csv('data.csv')
# display the datafram
df
```

Out[ ]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	
...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

569 rows × 33 columns

In [ ]:
```
# checking the dimensions of the dataset
print(f"Dataset contains {df.shape[0]} rows and {df.shape[1]} columns")
```

```
Dataset contains 569 rows and 33 columns
```

In [ ]:
```
# summary statistics of the dataset
df.describe()
```

Out[ ]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_r
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.04
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.03
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.00
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.02
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.03
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.07
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.20

8 rows × 32 columns

In [ ]:
```
# checking the data types
df.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
```

In [ ]:

```
# checking for missing values
missing_value = df.isnull().sum()
missing_values = missing_value[missing_value > 0]
print(f"""
Rows with Missing values in the dataset:
{missing_values}
""")
```

```
Rows with Missing values in the dataset:
Unnamed: 32    569
```

```
dtype: int64
```

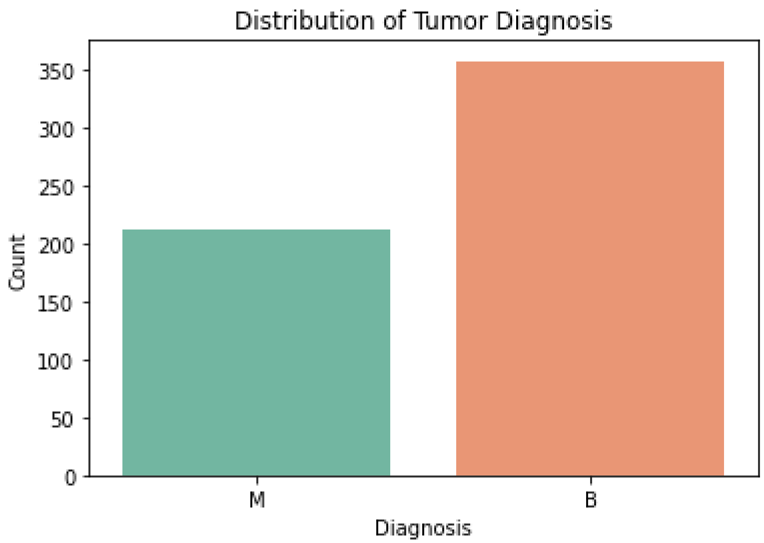
In [ ]:
```
# exploring the distribution of the target variable
df['diagnosis'].value_counts()
```

Out[ ]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

Distribution of Tumor Diagnosis

The bar chart below shows the distribution of the target variable, 'diagnosis', which indicates whether a tumor is Benign(B) or malignant(M).

In [ ]:
```
# visualizing the target variable
sns.countplot(x='diagnosis', data=df, palette='Set2')
plt.title('Distribution of Tumor Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Count')
plt.show()
```



Data Preparation

Preparing data for modelling by:

1. **Data cleaning**
2. **Feature selection**

3. **Feature scaling**
4. **Splitting the data**

Data Cleaning

In [ ]:
```
# dropping the column with missing values
df = df.drop('Unnamed: 32', axis=1)
```

In [ ]:
```
# rechecking for missing values
missing_val = df.isnull().sum()
if missing_val.sum() > 0:
    print("There are missing values")
else:
    print("There are no missing values")
```

```
There are no missing values
```

In [ ]:
```
# removing the ID column since it is not useful for modelling
df = df.drop(columns=['id'])
```

In [ ]:
```
# converting the diagnosis column to a numerical format
df['diagnosis'] = df['diagnosis'].map({'B':0, 'M':1})
```

In [ ]:
```
# saving the clean data to a new csv file
df.to_csv('cleaned_data.csv', index=False)
```

Feature Selection

In [ ]:
```
# selecting the target and the feature variables
y = df['diagnosis']
X = df.drop('diagnosis', axis=1)
```

Feature Scaling

In [ ]:
```
# standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Splitting the Data

In [ ]:
```
# splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, random_state=42)
# display train and test sizes
```

```
print(f"Training set size: {X_train.shape[0]}")
print(f"Testing set size: {X_test.shape[0]}")
```

```
Training set size: 426
Testing set size: 143
```

Modeling

1. **Train a Basic Logistic Regression Model**

In [ ]:
```
# instatiating the model
lr = LogisticRegression(random_state=42)
```

In [ ]:
```
# Train the model
lr.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(random_state=42)
```

In [ ]:
```
# making predictions
y_pred_lr = lr.predict(X_test)
```

In [ ]:
```
# Evaluating the model
print("Logistic Regression - Confusion MatriX: \n", confusion_matrix(y_test, y_pred_lr))
print("Logistic Regression - Classification Report: \n", classification_report(y_test, y_pred_lr))
```

```
Logistic Regression - Confusion MatriX:
 [[87  2]
 [ 1 53]]
Logistic Regression - Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.98      0.98        89
           1       0.96      0.98      0.97        54

    accuracy                           0.98       143
   macro avg       0.98      0.98      0.98       143
weighted avg       0.98      0.98      0.98       143
```

1. **Explore nonparametric models**

- **Decision Trees Model**

In [ ]:
```
# instatiating the model
dt = DecisionTreeClassifier(random_state=42)
```

In [ ]:
```
# training the model
dt.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeClassifier(random_state=42)
```

In [ ]:
```
# making predictions
y_pred_dt = dt.predict(X_test)
```

In [ ]:
```
# Evaluate the model
print("Decision Tree - Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("\nDecision Tree - Classification Report:\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree - Confusion Matrix:
 [[85  4]
 [ 3 51]]

Decision Tree - Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.96      0.96        89
           1       0.93      0.94      0.94        54

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

- **Random Forest Model**

In [ ]:
```
# instatiating the model
rf = RandomForestClassifier(random_state=42)
```

In [ ]:
```
# training the model
rf.fit(X_train, y_train)
```

Out[ ]:
```
RandomForestClassifier(random_state=42)
```

In [ ]:
```
# making predictions
y_pred_rf = rf.predict(X_test)
```

In [ ]:
```
# evaluating the model
print("Random Forest - Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("\nRandom Forest - Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest - Confusion Matrix:
 [[87  2]
 [ 3 51]]

Random Forest - Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.98      0.97        89
           1       0.96      0.94      0.95        54

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.96       143
```

- **Gradient Boosting**

In [ ]:
```
# instatiating the model
gb = GradientBoostingClassifier(random_state=42)
```

In [ ]:
```
# training the model
gb.fit(X_train, y_train)
```

Out[ ]:
```
GradientBoostingClassifier(random_state=42)
```

In [ ]:
```
# making predictions
y_pred_gb = gb.predict(X_test)
```

In [ ]:
```
# evaluating the model
print("Gradient Boosting - Confusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
print("\nGradient Boosting - Classification Report:\n", classification_report(y_test, y_pred_gb))
```

```
Gradient Boosting - Confusion Matrix:
 [[86  3]
 [ 3 51]]

Gradient Boosting - Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97        89
           1       0.94      0.94      0.94        54
```

```
    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

Evaluation

I will compare the Logistic Regression, Decision Tree, Random Forest and Gradient Boosting models using the following evaluation metrics:

1. Accuracy
2. Precision
3. Recall
4. F1-Score
5. Confusion Matrix

1. **Accuracy**

Models	Accuracy
Logistics Regression	98%
Decision Tree	95%
Random Forest	97%
Gradient Boosting	96%

Logistic Regression achieved the highest accuracy of 98%, indicating it correctly predicts the outcome most often compared to other models.

1. **Precision**

Models	Benign(0)	Malignant(1)
Logistics Regression	99%	96%
Decision Tree	97%	93%
Random Forest	97%	96%
Gradient Boosting	97%	94%

Logistic Regression has the highest precision compared to other classes, especially for class 0 (Benign) with 99% precision, meaning it has the fewest false positives.

1. **Recall**

Models	Benign(0)	Malignant(1)
Logistics Regression	98%	98%
Decision Tree	96%	94%
Random Forest	98%	94%
Gradient Boosting	97%	94%

Logistic Regression demonstrates better recall. Identifyies more true positives and having fewer false negatives.

1. **F1-Score**

Models	Benign(0)	Malignant(1)
Logistics Regression	98%	97%
Decision Tree	96%	94%
Random Forest	97%	95%
Gradient Boosting	97%	94%

Logistic Regression has higher F1-Scores. This indicates a better balance between precision and recall

1. **Confusion Matrix**

- Logistic Regression

	Actual Positive	Actual Negative
Predicted Positive	87	2
Predicted Negative	1	53

- Decision Tree

	Actual Positive	Actual Negative
Predicted Positive	85	4
Predicted Negative	3	51

- Random Forest

	Actual Positive	Actual Negative
Predicted Positive	87	2
Predicted Negative	3	51

- Gradient Boosting

	Actual Positive	Actual Negative
Predicted Positive	86	3
Predicted Negative	3	51

True Positives and True Negatives: Logistic regression correctly identifies more true positives and true negatives compared to other models.

False Positives and False Negatives: Logistic regression has fewer false positives and false negatives compared to other models.

The objective of this project was to develop a predictuve model to classify breast tumors as eithe benign or malignant using machine learning techniques.

After comparing multiple models, including:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Gradient Boosting

Logistic Regression emerged as the most effective model.

Feature Engeneering

Polynomial Features

In [ ]:
```
# Create polynomial features
poly = PolynomialFeatures(degree=2, include_bias=False, interaction_only=False)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

- I used `PolynomialFeatures` with a degree of 2 to create new features.

In [ ]:
```
# Feature scaling
scaler = StandardScaler()
X_train_poly = scaler.fit_transform(X_train_poly)
X_test_poly = scaler.transform(X_test_poly)
```

- I applied standard scaling using `StandardScaler` to ensure that all features have a mean of 0 and a standard deviation of 1.

In [ ]:
```
# Train logistic regression model with polynomial features
lg = LogisticRegression(max_iter=1000)
lg.fit(X_train_poly, y_train)
```

Out[ ]:
```
LogisticRegression(max_iter=1000)
```

- After transforming the features, I trained a logistic regression model on the new polynomial features.
- I trained the model with `max_iter=1000` to ensure convergence given the increased complexity of the feature set.

In [ ]:
```
# Predictions
y_pred = lg.predict(X_test_poly)
```

In [ ]:
```
# Get the names of the original features
feature_names = X.columns
feature_names
```

Out[ ]:
```
Index(['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'],
      dtype='object')
```

In [ ]:

```
# Get the names of the polynomial features
poly_feature_names = poly.get_feature_names(feature_names)
poly_feature_names
```

Out[ ]:

```
['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',
 'radius_mean^2',
 'radius_mean texture_mean',
 'radius_mean perimeter_mean',
 'radius_mean area_mean',
 'radius_mean smoothness_mean',
 'radius_mean compactness_mean',
 'radius_mean concavity_mean',
 'radius_mean concave points_mean',
 'radius_mean symmetry_mean',
 'radius_mean fractal_dimension_mean',
 'radius_mean radius_se',
```

```
            'radius_mean texture_se',
            'radius_mean perimeter_se',
            'radius_mean area_se',
            'radius_mean smoothness_se',
            'radius_mean compactness_se',
            'radius_mean concavity_se',
            'radius_mean concave points_se',
            'radius_mean symmetry_se',
            'radius_mean fractal_dimension_se',
            'radius_mean radius_worst',
            'radius_mean texture_worst',
            'radius_mean perimeter_worst',
            'radius_mean area_worst',
            'radius_mean smoothness_worst',
            'radius_mean compactness_worst',
            'radius_mean concavity_worst',
            'radius_mean concave points_worst',
            'radius_mean symmetry_worst',
            'radius_mean fractal_dimension_worst',
            'texture_mean^2',
            'texture_mean perimeter_mean',
            'texture_mean area_mean',
            'texture_mean smoothness_mean',
            'texture_mean compactness_mean',
            'texture_mean concavity_mean',
            'texture_mean concave points_mean',
            'texture_mean symmetry_mean',
            'texture_mean fractal_dimension_mean',
            'texture_mean radius_se',
            'texture_mean texture_se',
            'texture_mean perimeter_se',
            'texture_mean area_se',
            'texture_mean smoothness_se',
            'texture_mean compactness_se',
            'texture_mean concavity_se',
            'texture_mean concave points_se',
            'texture_mean symmetry_se',
            'texture_mean fractal_dimension_se',
            'texture_mean radius_worst',
            'texture_mean texture_worst',
            'texture_mean perimeter_worst',
            'texture_mean area_worst',
            'texture_mean smoothness_worst',
            'texture_mean compactness_worst',
            'texture_mean concavity_worst',
            'texture_mean concave points_worst',
            'texture_mean symmetry_worst',
            'texture_mean fractal_dimension_worst',
            'perimeter_mean^2',
```

```
'perimeter_mean area_mean',
'perimeter_mean smoothness_mean',
'perimeter_mean compactness_mean',
'perimeter_mean concavity_mean',
'perimeter_mean concave points_mean',
'perimeter_mean symmetry_mean',
'perimeter_mean fractal_dimension_mean',
'perimeter_mean radius_se',
'perimeter_mean texture_se',
'perimeter_mean perimeter_se',
'perimeter_mean area_se',
'perimeter_mean smoothness_se',
'perimeter_mean compactness_se',
'perimeter_mean concavity_se',
'perimeter_mean concave points_se',
'perimeter_mean symmetry_se',
'perimeter_mean fractal_dimension_se',
'perimeter_mean radius_worst',
'perimeter_mean texture_worst',
'perimeter_mean perimeter_worst',
'perimeter_mean area_worst',
'perimeter_mean smoothness_worst',
'perimeter_mean compactness_worst',
'perimeter_mean concavity_worst',
'perimeter_mean concave points_worst',
'perimeter_mean symmetry_worst',
'perimeter_mean fractal_dimension_worst',
'area_mean^2',
'area_mean smoothness_mean',
'area_mean compactness_mean',
'area_mean concavity_mean',
'area_mean concave points_mean',
'area_mean symmetry_mean',
'area_mean fractal_dimension_mean',
'area_mean radius_se',
'area_mean texture_se',
'area_mean perimeter_se',
'area_mean area_se',
'area_mean smoothness_se',
'area_mean compactness_se',
'area_mean concavity_se',
'area_mean concave points_se',
'area_mean symmetry_se',
'area_mean fractal_dimension_se',
'area_mean radius_worst',
'area_mean texture_worst',
'area_mean perimeter_worst',
'area_mean area_worst',
'area_mean smoothness_worst',
```

```
               'area_mean compactness_worst',
               'area_mean concavity_worst',
               'area_mean concave points_worst',
               'area_mean symmetry_worst',
               'area_mean fractal_dimension_worst',
               'smoothness_mean^2',
               'smoothness_mean compactness_mean',
               'smoothness_mean concavity_mean',
               'smoothness_mean concave points_mean',
               'smoothness_mean symmetry_mean',
               'smoothness_mean fractal_dimension_mean',
               'smoothness_mean radius_se',
               'smoothness_mean texture_se',
               'smoothness_mean perimeter_se',
               'smoothness_mean area_se',
               'smoothness_mean smoothness_se',
               'smoothness_mean compactness_se',
               'smoothness_mean concavity_se',
               'smoothness_mean concave points_se',
               'smoothness_mean symmetry_se',
               'smoothness_mean fractal_dimension_se',
               'smoothness_mean radius_worst',
               'smoothness_mean texture_worst',
               'smoothness_mean perimeter_worst',
               'smoothness_mean area_worst',
               'smoothness_mean smoothness_worst',
               'smoothness_mean compactness_worst',
               'smoothness_mean concavity_worst',
               'smoothness_mean concave points_worst',
               'smoothness_mean symmetry_worst',
               'smoothness_mean fractal_dimension_worst',
               'compactness_mean^2',
               'compactness_mean concavity_mean',
               'compactness_mean concave points_mean',
               'compactness_mean symmetry_mean',
               'compactness_mean fractal_dimension_mean',
               'compactness_mean radius_se',
               'compactness_mean texture_se',
               'compactness_mean perimeter_se',
               'compactness_mean area_se',
               'compactness_mean smoothness_se',
               'compactness_mean compactness_se',
               'compactness_mean concavity_se',
               'compactness_mean concave points_se',
               'compactness_mean symmetry_se',
               'compactness_mean fractal_dimension_se',
               'compactness_mean radius_worst',
               'compactness_mean texture_worst',
               'compactness_mean perimeter_worst',
```

```
'compactness_mean area_worst',
'compactness_mean smoothness_worst',
'compactness_mean compactness_worst',
'compactness_mean concavity_worst',
'compactness_mean concave points_worst',
'compactness_mean symmetry_worst',
'compactness_mean fractal_dimension_worst',
'concavity_mean^2',
'concavity_mean concave points_mean',
'concavity_mean symmetry_mean',
'concavity_mean fractal_dimension_mean',
'concavity_mean radius_se',
'concavity_mean texture_se',
'concavity_mean perimeter_se',
'concavity_mean area_se',
'concavity_mean smoothness_se',
'concavity_mean compactness_se',
'concavity_mean concavity_se',
'concavity_mean concave points_se',
'concavity_mean symmetry_se',
'concavity_mean fractal_dimension_se',
'concavity_mean radius_worst',
'concavity_mean texture_worst',
'concavity_mean perimeter_worst',
'concavity_mean area_worst',
'concavity_mean smoothness_worst',
'concavity_mean compactness_worst',
'concavity_mean concavity_worst',
'concavity_mean concave points_worst',
'concavity_mean symmetry_worst',
'concavity_mean fractal_dimension_worst',
'concave points_mean^2',
'concave points_mean symmetry_mean',
'concave points_mean fractal_dimension_mean',
'concave points_mean radius_se',
'concave points_mean texture_se',
'concave points_mean perimeter_se',
'concave points_mean area_se',
'concave points_mean smoothness_se',
'concave points_mean compactness_se',
'concave points_mean concavity_se',
'concave points_mean concave points_se',
'concave points_mean symmetry_se',
'concave points_mean fractal_dimension_se',
'concave points_mean radius_worst',
'concave points_mean texture_worst',
'concave points_mean perimeter_worst',
'concave points_mean area_worst',
'concave points_mean smoothness_worst',
```

```
                'concave points_mean compactness_worst',
                'concave points_mean concavity_worst',
                'concave points_mean concave points_worst',
                'concave points_mean symmetry_worst',
                'concave points_mean fractal_dimension_worst',
                'symmetry_mean^2',
                'symmetry_mean fractal_dimension_mean',
                'symmetry_mean radius_se',
                'symmetry_mean texture_se',
                'symmetry_mean perimeter_se',
                'symmetry_mean area_se',
                'symmetry_mean smoothness_se',
                'symmetry_mean compactness_se',
                'symmetry_mean concavity_se',
                'symmetry_mean concave points_se',
                'symmetry_mean symmetry_se',
                'symmetry_mean fractal_dimension_se',
                'symmetry_mean radius_worst',
                'symmetry_mean texture_worst',
                'symmetry_mean perimeter_worst',
                'symmetry_mean area_worst',
                'symmetry_mean smoothness_worst',
                'symmetry_mean compactness_worst',
                'symmetry_mean concavity_worst',
                'symmetry_mean concave points_worst',
                'symmetry_mean symmetry_worst',
                'symmetry_mean fractal_dimension_worst',
                'fractal_dimension_mean^2',
                'fractal_dimension_mean radius_se',
                'fractal_dimension_mean texture_se',
                'fractal_dimension_mean perimeter_se',
                'fractal_dimension_mean area_se',
                'fractal_dimension_mean smoothness_se',
                'fractal_dimension_mean compactness_se',
                'fractal_dimension_mean concavity_se',
                'fractal_dimension_mean concave points_se',
                'fractal_dimension_mean symmetry_se',
                'fractal_dimension_mean fractal_dimension_se',
                'fractal_dimension_mean radius_worst',
                'fractal_dimension_mean texture_worst',
                'fractal_dimension_mean perimeter_worst',
                'fractal_dimension_mean area_worst',
                'fractal_dimension_mean smoothness_worst',
                'fractal_dimension_mean compactness_worst',
                'fractal_dimension_mean concavity_worst',
                'fractal_dimension_mean concave points_worst',
                'fractal_dimension_mean symmetry_worst',
                'fractal_dimension_mean fractal_dimension_worst',
                'radius_se^2',
```

```
'radius_se texture_se',
'radius_se perimeter_se',
'radius_se area_se',
'radius_se smoothness_se',
'radius_se compactness_se',
'radius_se concavity_se',
'radius_se concave points_se',
'radius_se symmetry_se',
'radius_se fractal_dimension_se',
'radius_se radius_worst',
'radius_se texture_worst',
'radius_se perimeter_worst',
'radius_se area_worst',
'radius_se smoothness_worst',
'radius_se compactness_worst',
'radius_se concavity_worst',
'radius_se concave points_worst',
'radius_se symmetry_worst',
'radius_se fractal_dimension_worst',
'texture_se^2',
'texture_se perimeter_se',
'texture_se area_se',
'texture_se smoothness_se',
'texture_se compactness_se',
'texture_se concavity_se',
'texture_se concave points_se',
'texture_se symmetry_se',
'texture_se fractal_dimension_se',
'texture_se radius_worst',
'texture_se texture_worst',
'texture_se perimeter_worst',
'texture_se area_worst',
'texture_se smoothness_worst',
'texture_se compactness_worst',
'texture_se concavity_worst',
'texture_se concave points_worst',
'texture_se symmetry_worst',
'texture_se fractal_dimension_worst',
'perimeter_se^2',
'perimeter_se area_se',
'perimeter_se smoothness_se',
'perimeter_se compactness_se',
'perimeter_se concavity_se',
'perimeter_se concave points_se',
'perimeter_se symmetry_se',
'perimeter_se fractal_dimension_se',
'perimeter_se radius_worst',
'perimeter_se texture_worst',
'perimeter_se perimeter_worst',
```

```
 'perimeter_se area_worst',
 'perimeter_se smoothness_worst',
 'perimeter_se compactness_worst',
 'perimeter_se concavity_worst',
 'perimeter_se concave points_worst',
 'perimeter_se symmetry_worst',
 'perimeter_se fractal_dimension_worst',
 'area_se^2',
 'area_se smoothness_se',
 'area_se compactness_se',
 'area_se concavity_se',
 'area_se concave points_se',
 'area_se symmetry_se',
 'area_se fractal_dimension_se',
 'area_se radius_worst',
 'area_se texture_worst',
 'area_se perimeter_worst',
 'area_se area_worst',
 'area_se smoothness_worst',
 'area_se compactness_worst',
 'area_se concavity_worst',
 'area_se concave points_worst',
 'area_se symmetry_worst',
 'area_se fractal_dimension_worst',
 'smoothness_se^2',
 'smoothness_se compactness_se',
 'smoothness_se concavity_se',
 'smoothness_se concave points_se',
 'smoothness_se symmetry_se',
 'smoothness_se fractal_dimension_se',
 'smoothness_se radius_worst',
 'smoothness_se texture_worst',
 'smoothness_se perimeter_worst',
 'smoothness_se area_worst',
 'smoothness_se smoothness_worst',
 'smoothness_se compactness_worst',
 'smoothness_se concavity_worst',
 'smoothness_se concave points_worst',
 'smoothness_se symmetry_worst',
 'smoothness_se fractal_dimension_worst',
 'compactness_se^2',
 'compactness_se concavity_se',
 'compactness_se concave points_se',
 'compactness_se symmetry_se',
 'compactness_se fractal_dimension_se',
 'compactness_se radius_worst',
 'compactness_se texture_worst',
 'compactness_se perimeter_worst',
 'compactness_se area_worst',
```

```
               'compactness_se smoothness_worst',
               'compactness_se compactness_worst',
               'compactness_se concavity_worst',
               'compactness_se concave points_worst',
               'compactness_se symmetry_worst',
               'compactness_se fractal_dimension_worst',
               'concavity_se^2',
               'concavity_se concave points_se',
               'concavity_se symmetry_se',
               'concavity_se fractal_dimension_se',
               'concavity_se radius_worst',
               'concavity_se texture_worst',
               'concavity_se perimeter_worst',
               'concavity_se area_worst',
               'concavity_se smoothness_worst',
               'concavity_se compactness_worst',
               'concavity_se concavity_worst',
               'concavity_se concave points_worst',
               'concavity_se symmetry_worst',
               'concavity_se fractal_dimension_worst',
               'concave points_se^2',
               'concave points_se symmetry_se',
               'concave points_se fractal_dimension_se',
               'concave points_se radius_worst',
               'concave points_se texture_worst',
               'concave points_se perimeter_worst',
               'concave points_se area_worst',
               'concave points_se smoothness_worst',
               'concave points_se compactness_worst',
               'concave points_se concavity_worst',
               'concave points_se concave points_worst',
               'concave points_se symmetry_worst',
               'concave points_se fractal_dimension_worst',
               'symmetry_se^2',
               'symmetry_se fractal_dimension_se',
               'symmetry_se radius_worst',
               'symmetry_se texture_worst',
               'symmetry_se perimeter_worst',
               'symmetry_se area_worst',
               'symmetry_se smoothness_worst',
               'symmetry_se compactness_worst',
               'symmetry_se concavity_worst',
               'symmetry_se concave points_worst',
               'symmetry_se symmetry_worst',
               'symmetry_se fractal_dimension_worst',
               'fractal_dimension_se^2',
               'fractal_dimension_se radius_worst',
               'fractal_dimension_se texture_worst',
               'fractal_dimension_se perimeter_worst',
```

```
'fractal_dimension_se area_worst',
'fractal_dimension_se smoothness_worst',
'fractal_dimension_se compactness_worst',
'fractal_dimension_se concavity_worst',
'fractal_dimension_se concave points_worst',
'fractal_dimension_se symmetry_worst',
'fractal_dimension_se fractal_dimension_worst',
'radius_worst^2',
'radius_worst texture_worst',
'radius_worst perimeter_worst',
'radius_worst area_worst',
'radius_worst smoothness_worst',
'radius_worst compactness_worst',
'radius_worst concavity_worst',
'radius_worst concave points_worst',
'radius_worst symmetry_worst',
'radius_worst fractal_dimension_worst',
'texture_worst^2',
'texture_worst perimeter_worst',
'texture_worst area_worst',
'texture_worst smoothness_worst',
'texture_worst compactness_worst',
'texture_worst concavity_worst',
'texture_worst concave points_worst',
'texture_worst symmetry_worst',
'texture_worst fractal_dimension_worst',
'perimeter_worst^2',
'perimeter_worst area_worst',
'perimeter_worst smoothness_worst',
'perimeter_worst compactness_worst',
'perimeter_worst concavity_worst',
'perimeter_worst concave points_worst',
'perimeter_worst symmetry_worst',
'perimeter_worst fractal_dimension_worst',
'area_worst^2',
'area_worst smoothness_worst',
'area_worst compactness_worst',
'area_worst concavity_worst',
'area_worst concave points_worst',
'area_worst symmetry_worst',
'area_worst fractal_dimension_worst',
'smoothness_worst^2',
'smoothness_worst compactness_worst',
'smoothness_worst concavity_worst',
'smoothness_worst concave points_worst',
'smoothness_worst symmetry_worst',
'smoothness_worst fractal_dimension_worst',
'compactness_worst^2',
'compactness_worst concavity_worst',
```

```
 'compactness_worst concave points_worst',
 'compactness_worst symmetry_worst',
 'compactness_worst fractal_dimension_worst',
 'concavity_worst^2',
 'concavity_worst concave points_worst',
 'concavity_worst symmetry_worst',
 'concavity_worst fractal_dimension_worst',
 'concave points_worst^2',
 'concave points_worst symmetry_worst',
 'concave points_worst fractal_dimension_worst',
 'symmetry_worst^2',
 'symmetry_worst fractal_dimension_worst',
 'fractal_dimension_worst^2']
```

In [ ]:
```python
# Create a DataFrame to view the polynomial features
X_train_poly_df = pd.DataFrame(X_train_poly, columns=poly_feature_names)
X_train_poly_df.head()
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mea
0	-0.349138	-1.438513	-0.411726	-0.390479	-1.863662	-1.268607	-0.826171	-0.952866	-1.7293
1	-0.204687	0.312640	-0.133673	-0.275880	1.078073	0.863546	0.726314	0.898441	1.1787
2	-0.329312	-0.215072	-0.317394	-0.364357	-1.579880	-0.457451	-0.597310	-0.764588	0.2753
3	1.027403	2.089824	1.046922	0.917584	0.316303	0.562037	1.048527	0.930437	-0.3256
4	1.828969	0.696001	1.763681	1.783821	-0.333674	0.628175	0.974660	1.265740	-0.1315

5 rows × 495 columns

In [ ]:
```python
# Coefficients of the logistic regression model
coefficients = lg.coef_[0]
```

In [ ]:
```python
# Combine feature names and their corresponding coefficients
feature_importance = pd.DataFrame({
    'Feature': poly_feature_names,
    'Coefficient': coefficients
})
```

In [ ]:
```
# Sort by absolute value of the coefficient
feature_importance['Importance'] = np.abs(feature_importance['Coefficient'])
feature_importance.sort_values(by='Importance', ascending=False, inplace=True)
```

In [ ]:
```
# Display the top features
feature_importance.head(15)
```

Out[ ]:

	Feature	Coefficient	Importance
21	texture_worst	1.124504	1.124504
26	concavity_worst	0.889373	0.889373
20	radius_worst	0.863021	0.863021
27	concave points_worst	0.848735	0.848735
7	concave points_mean	0.803998	0.803998
23	area_worst	0.795675	0.795675
22	perimeter_worst	0.691173	0.691173
24	smoothness_worst	0.690765	0.690765
15	compactness_se	-0.640389	0.640389
1	texture_mean	0.639375	0.639375
18	symmetry_se	-0.634214	0.634214
6	concavity_mean	0.633679	0.633679
10	radius_se	0.592835	0.592835
0	radius_mean	0.519386	0.519386
13	area_se	0.509549	0.509549

- The addition of polynomial features allows the logistic regression model to better capture non-linear patterns in the data, potentially leading to improved classification performance.

VISUALIZATIONS

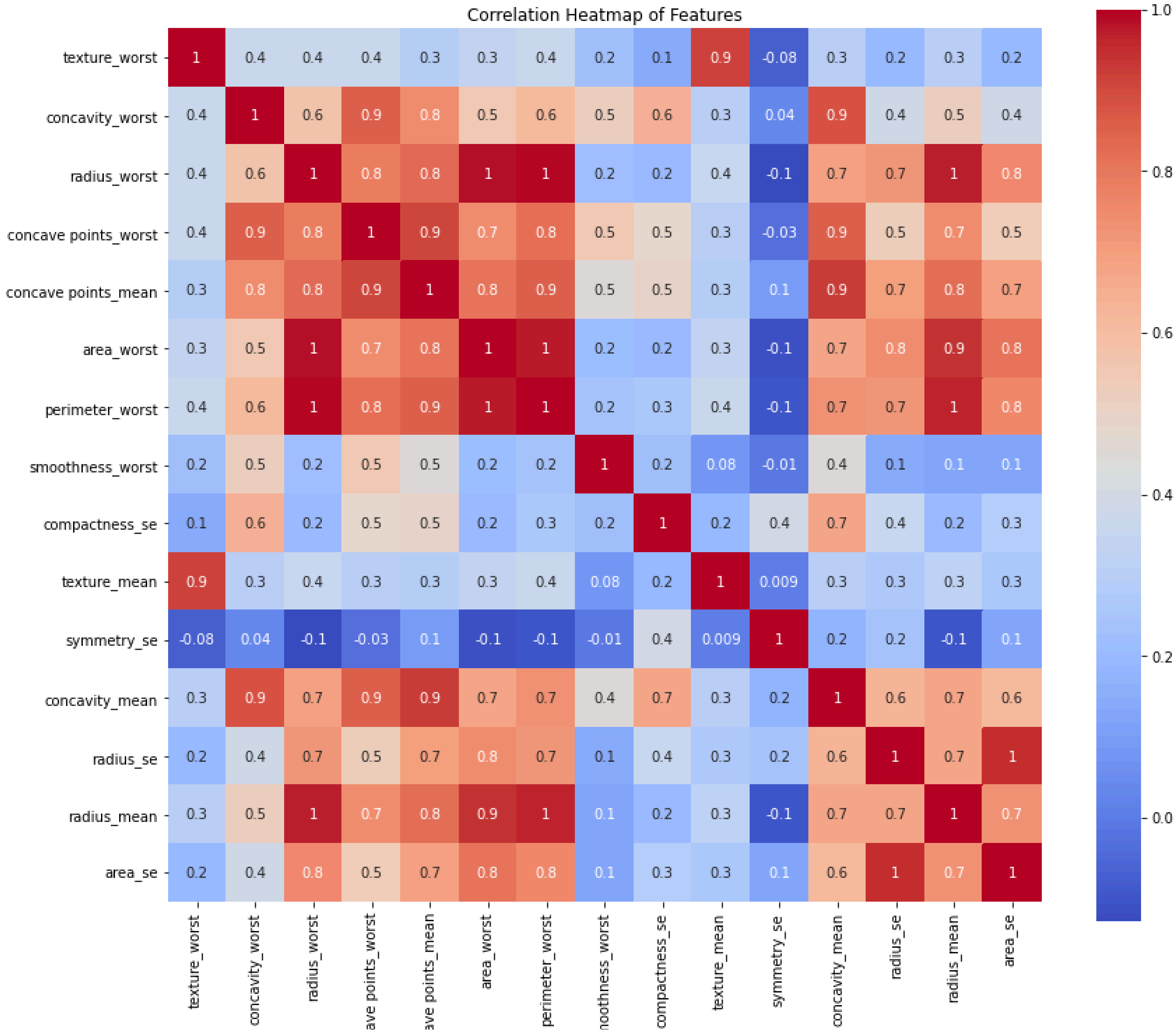
selecting features to use for visualization

In [ ]:
```
Top_features = ['texture_worst', 'concavity_worst', 'radius_worst', 'concave points_worst', 'concave points_mean', 'area_
```

1. **Correlation Heatmap**

The heatmap below illustrates the correlation between different features in the dataset. Strongly correlated features can provide insights into the relationships between variables and help in feature selection.

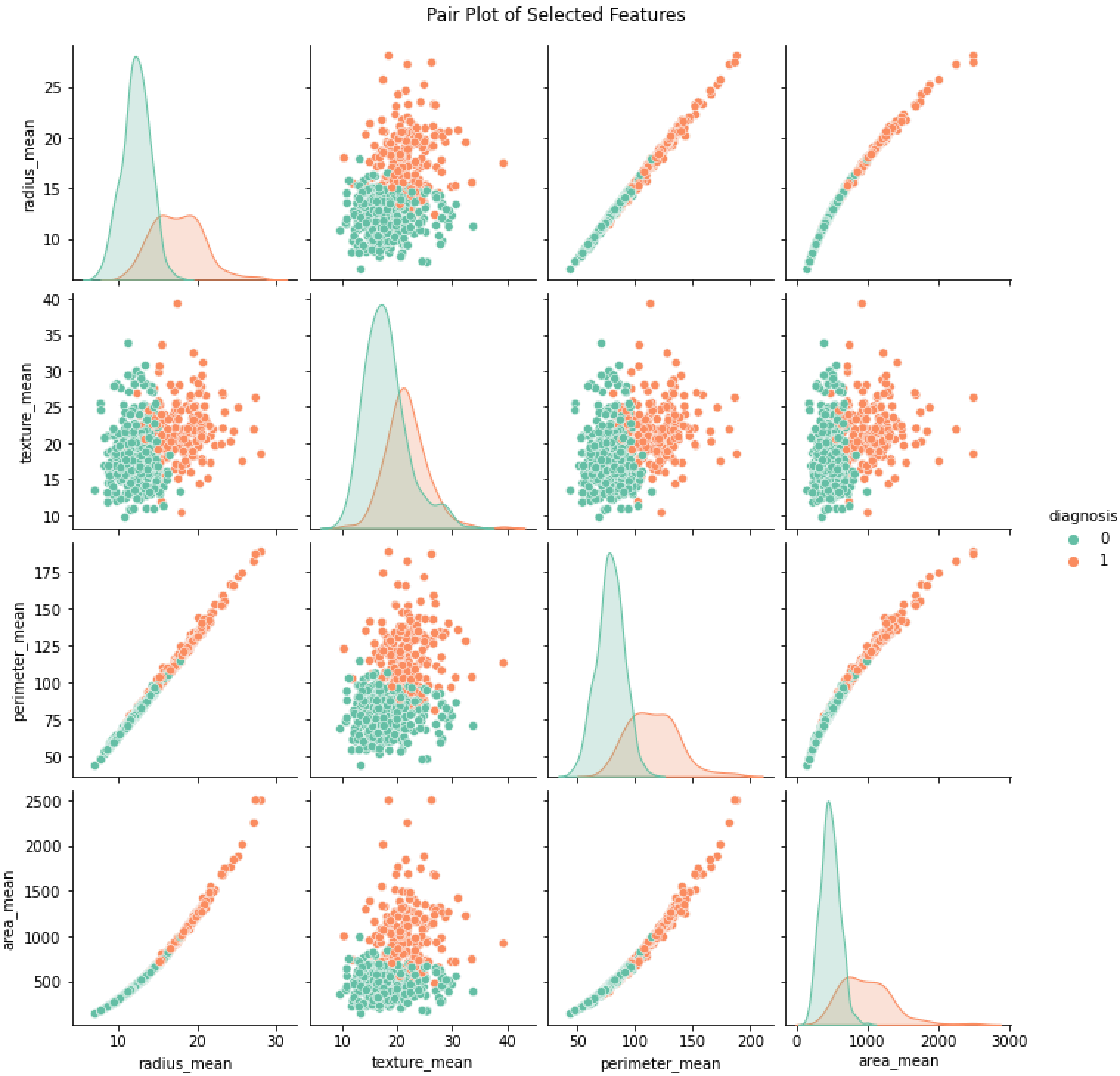
In [ ]:
```
# Correlation heatmap to understand relationships between features
plt.figure(figsize=(14, 12))
sns.heatmap(df[Top_features].corr(), annot=True, square=True, cmap='coolwarm', fmt= '.0g')
plt.title('Correlation Heatmap of Features')
plt.show()
```


Correlation Heatmap of Features
texture_worst
concavity_worst
radius_worst
concave points_worst
concave points_mean
area_worst
perimeter_worst
smoothness_worst
compactness_se
texture_mean
symmetry_se
concavity_mean
radius_se
radius_mean
area_se
1 0.4 0.4 0.4 0.3 0.3 0.4 0.2 0.1 0.9 -0.08 0.3 0.2 0.3 0.2
0.4 1 0.6 0.9 0.8 0.5 0.6 0.5 0.6 0.3 0.04 0.9 0.4 0.5 0.4
0.4 0.6 1 0.8 0.8 1 1 0.2 0.2 0.4 -0.1 0.7 0.7 1 0.8
0.4 0.9 0.8 1 0.9 0.7 0.8 0.5 0.5 0.3 -0.03 0.9 0.5 0.7 0.5
0.3 0.8 0.8 0.9 1 0.8 0.9 0.5 0.5 0.3 0.1 0.9 0.7 0.8 0.7
0.3 0.5 1 0.7 0.8 1 1 0.2 0.2 0.3 -0.1 0.7 0.8 0.9 0.8
0.4 0.6 1 0.8 0.9 1 1 0.2 0.3 0.4 -0.1 0.7 0.7 1 0.8
0.2 0.5 0.2 0.5 0.5 0.2 0.2 1 0.2 0.08 -0.01 0.4 0.1 0.1 0.1
0.1 0.6 0.2 0.5 0.5 0.2 0.3 0.2 1 0.2 0.4 0.7 0.4 0.2 0.3
0.9 0.3 0.4 0.3 0.3 0.3 0.4 0.08 0.2 1 0.009 0.3 0.3 0.3 0.3
-0.08 0.04 -0.1 -0.03 0.1 -0.1 -0.1 -0.01 0.4 0.009 1 0.2 0.2 -0.1 0.1
0.3 0.9 0.7 0.9 0.9 0.7 0.7 0.4 0.7 0.3 0.2 1 0.6 0.7 0.6
0.2 0.4 0.7 0.5 0.7 0.8 0.7 0.1 0.4 0.3 0.2 0.6 1 0.7 1
0.3 0.5 1 0.7 0.8 0.9 1 0.1 0.2 0.3 -0.1 0.7 0.7 1 0.7
0.2 0.4 0.8 0.5 0.7 0.8 0.8 0.1 0.3 0.3 0.1 0.6 1 0.7 1
texture_worst
concavity_worst
radius_worst
ave points_worst
ave points_mean
area_worst
perimeter_worst
noothness_worst
compactness_se
texture_mean
symmetry_se
concavity_mean
radius_se
radius_mean
area_se
1.0
0.8
0.6
0.4
0.2
0.0

1. **Pair Plot of Selected Features**

The pair plot below shows the relationships between selected features (`mean_radius` , `mean_texture` , `mean_perimeter` , `mean_area`) and the target variable (`diagnosis`). This visualization helps to observe how different features differentiate benign and malignant tumors.

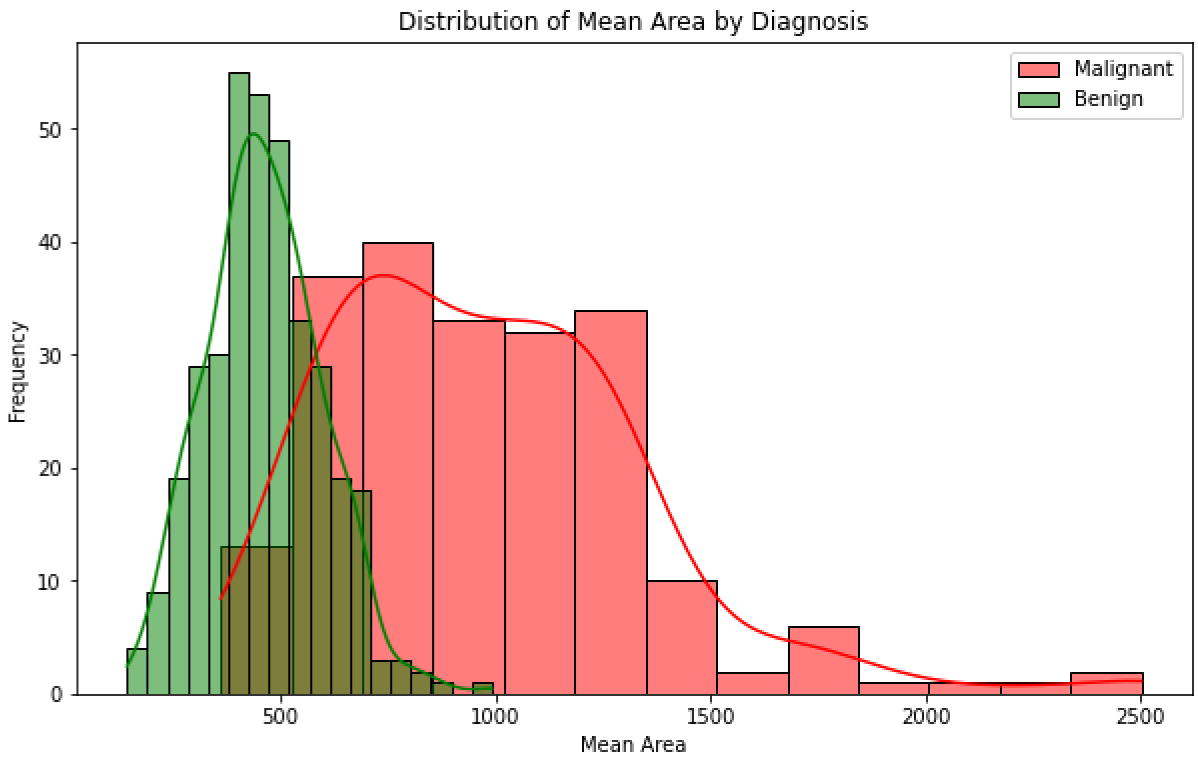
In [ ]:
```
# Pair plot for selected features
selected_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'diagnosis']
sns.pairplot(df[selected_features], hue='diagnosis', palette='Set2')
plt.suptitle('Pair Plot of Selected Features', y=1.02)
plt.show()
```


Pair Plot of Selected Features
radius_mean
texture_mean
perimeter_mean
area_mean
diagnosis
0
1

1. **Distribution of Feature Values for Each Diagnosis**

The histogram below shows the distribution of `area_mean` for benign and malignant tumors. Such plots help identify whether certain features have distinct ranges for different classes, which can aid in classification.

In [ ]:
```
# Distribution of a selected feature based on diagnosis
plt.figure(figsize=(10, 6))
sns.histplot(df[df['diagnosis'] == 1]['area_mean'], color='red', label='Malignant', kde=True)
sns.histplot(df[df['diagnosis'] == 0]['area_mean'], color='green', label='Benign', kde=True)
plt.title('Distribution of Mean Area by Diagnosis')
plt.xlabel('Mean Area')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

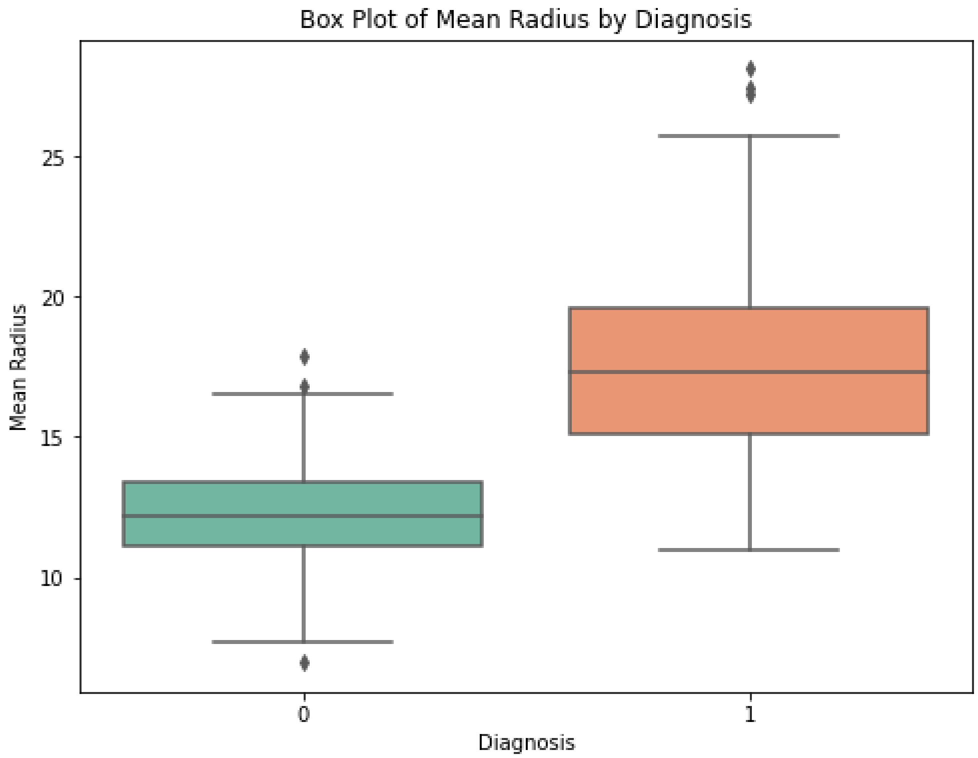


1. **Box Plot of Feature Values by Diagnosis**

The box plot below compares the `radius_mean` of tumors for benign and malignant diagnoses. This type of plot is useful for understanding the spread and central tendency of feature values across different classes.

In [ ]:

```
# Box plot for a selected feature based on diagnosis
plt.figure(figsize=(8, 6))
sns.boxplot(x='diagnosis', y='radius_mean', data=df, palette='Set2')
plt.title('Box Plot of Mean Radius by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Mean Radius')
plt.show()
```



Summary

The project successfully developed a reliable and interpretable model for breast cancer prediction, demonstrating the effectiveness of using machine learning techniques in healthcare applications. Implementing such models can significantly aid in early detection and treatment planning, ultimately contributing to better patient outcomes.