Breast Cancer Classification Prediction

A project by Rafif

Project Overview

Goal
To predict whether a breast cancer case is benign or malignant based on the available features.

Dataset
This dataset is obtained from the Breast Cancer Wisconsin Dataset (Kaggle), and it is also one of the standard datasets in Scikit-learn.

Focus
This project focuses on standard data cleaning and EDA pipeline, hyperparameter tuning using grid search, dimensionality reduction using PCA, and classification using logistic regression and SVM.

Dataset

Contains 33 columns:

- 1 ID column and 1 NaN column
- 30 features:
  - 10 unique features: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension
  - 3 measurements: mean, standard error, worst
- The target `diagnosis`: 357 benign, 212 malignant

10 'mean' columns

```
2   radius_mean
3   texture_mean
4   perimeter_mean
5   area_mean
6   smoothness_mean
7   compactness_mean
8   concavity_mean
9   concave points_mean
10  symmetry_mean
11  fractal_dimension_mean
```

10 'standard error' columns

```
12  radius_se
13  texture_se
14  perimeter_se
15  area_se
16  smoothness_se
17  compactness_se
18  concavity_se
19  concave points_se
20  symmetry_se
21  fractal_dimension_se
```

10 'worst' columns

```
22  radius_worst
23  texture_worst
24  perimeter_worst
25  area_worst
26  smoothness_worst
27  compactness_worst
28  concavity_worst
29  concave points_worst
30  symmetry_worst
31  fractal_dimension_worst
```

Features were directly computed from the cell nucleus

Preprocessing

Dropping unimportant columns: `id` and `Unnamed: 32`

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

No duplicated rows

```
df.duplicated().sum()
```

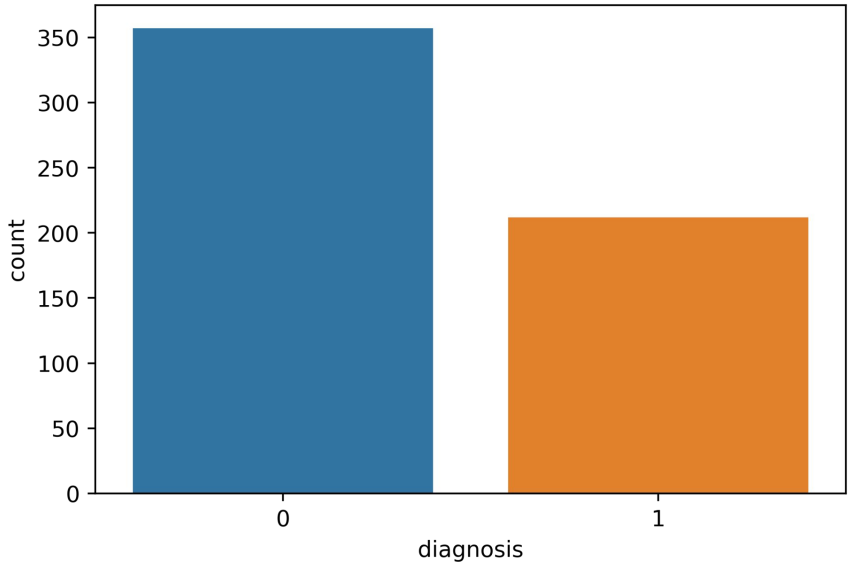
```
0
```

Binary encoding for `diagnosis`: benign (0), malignant (1)

```
# Binary encoding for target variable
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
```

EDA

Check the distribution of `diagnosis`



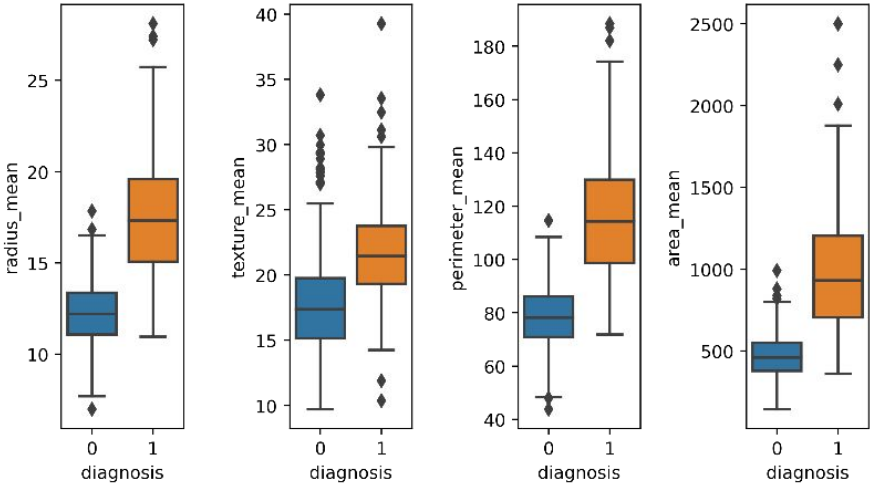
diagnosis	percentage
0	62.741652
1	37.258348

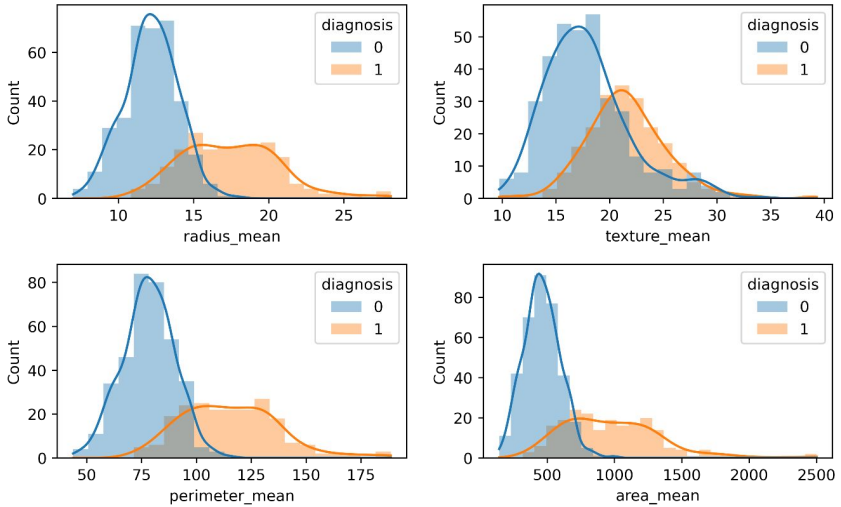
More benign cases (1.5 times) than malignant cases → severe cases are rarer

Slightly imbalanced, but not too extreme → balancing not needed

EDA

Visualize distribution and check outliers with boxplot and histogram





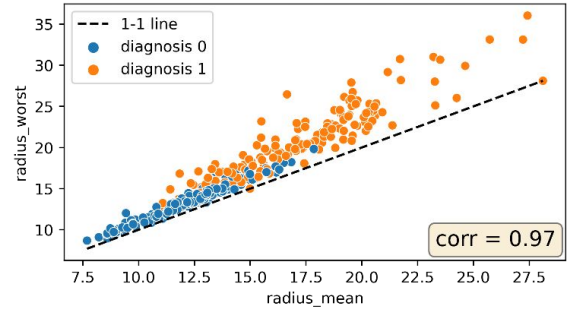
Malignant cases have systematically larger values for the features

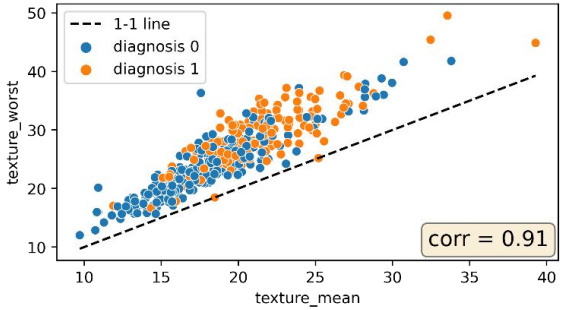
No extreme outliers in the dataset

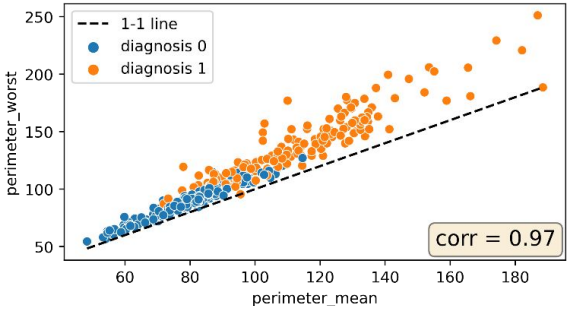
Feature Selection

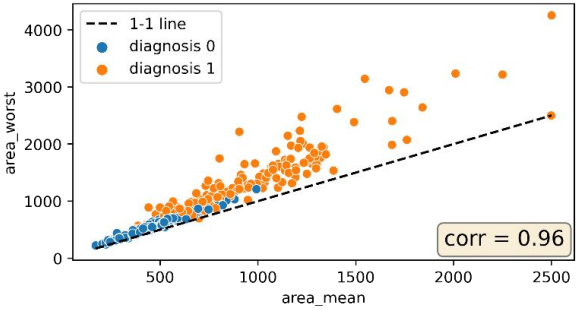
Train-test split to prevent information leakage (70:30 ratio)

Check the correlation between `_mean` and `_worst` features









High correlation (as expected)

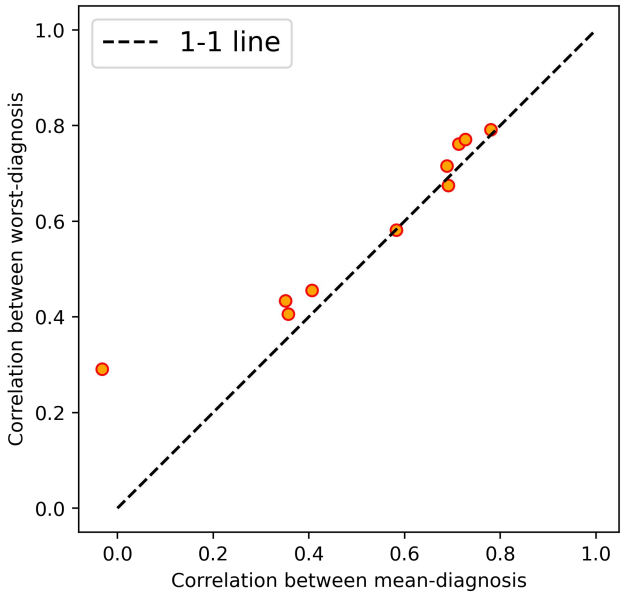
Have to pick one as predictors: `_mean` or `_worst`

Feature Selection

Pick the one with an overall higher correlation with `diagnosis`

Using point-biserial correlation (special case of Pearson correlation)

	features	_mean	_worst
0	radius	0.712933	0.761693
1	texture	0.406121	0.455530
2	perimeter	0.726666	0.770895
3	area	0.688087	0.715231
4	smoothness	0.357168	0.405991
5	compactness	0.582411	0.581583
6	concavity	0.691392	0.674592
7	concave points	0.780350	0.791089
8	symmetry	0.351107	0.433798
9	fractal	-0.032082	0.290516



Overall, `_worst` features have higher correlation with `diagnosis`

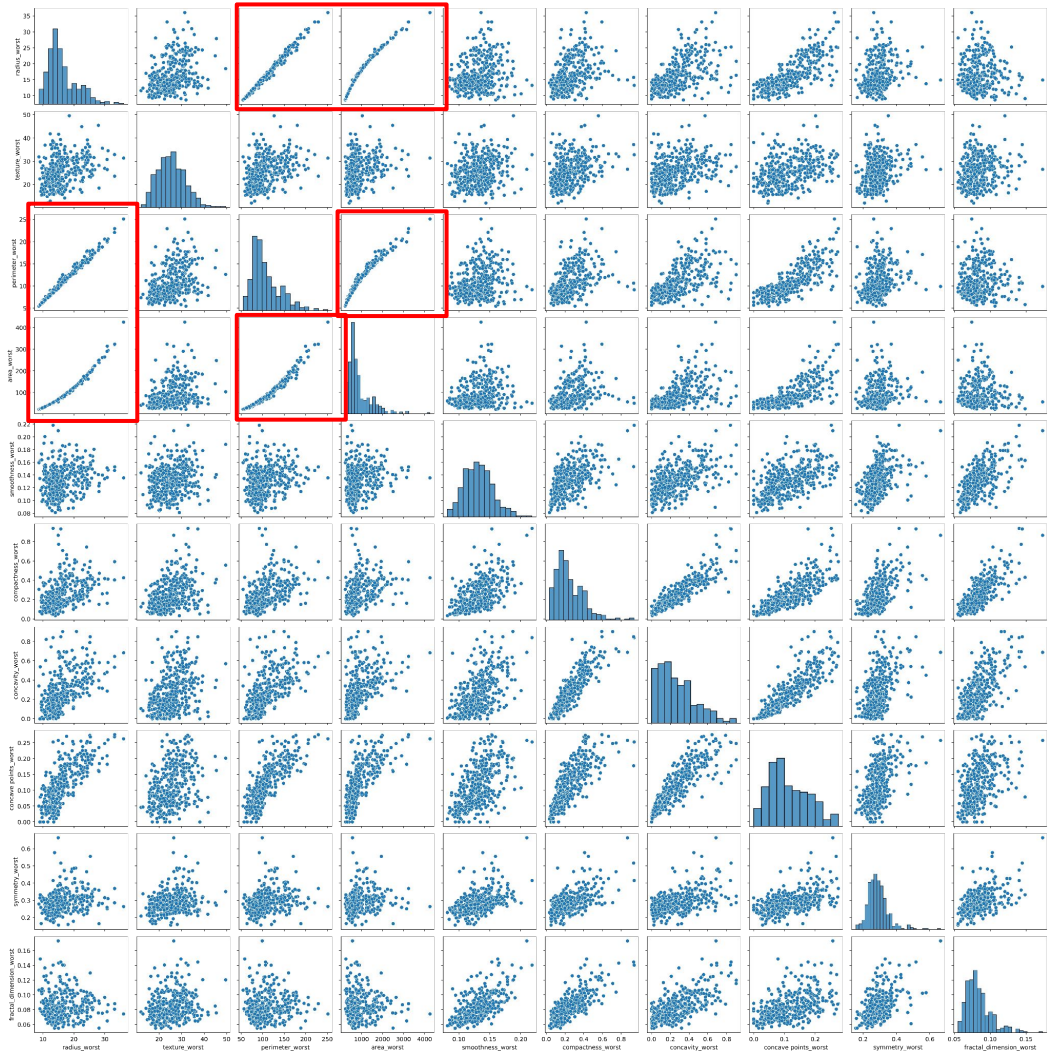
Drop all `_mean` features

Feature Selection

Check correlations within `_worst` features

`radius` is correlated with `perimeter` and `area` (also expected)

`perimeter` has the largest correlation with `perimeter` diagnosis → dropping `radius` and `area`



Building the Model: Logistic Regression

Linearly separable case → using logistic regression and support vector machine (SVM)

Grid search method to optimize the hyperparameters

```
# Import the model
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state=42,max_iter=1000)

# Import GridSearch
from sklearn.model_selection import GridSearchCV

# hyperparameter values to be tuned
param_grid = {'C': np.logspace(-2,2,5),
              'solver': ['newton-cg', 'lbfgs', 'liblinear'],
              'penalty': ['l1','l2']}

# the tuning
logreg_gridcv = GridSearchCV(estimator=logreg, param_grid=param_grid, cv=5,
                             scoring='recall',verbose=3)
logreg_gridcv.fit(X_train, y_train.values.ravel())
```

Best hyperparameters:

- C = 100
- Penalty = l2
- Solver = newton-cg

Evaluating the Model: Logistic Regression

Recall as metric → as few false negatives as possible

- Better to falsely diagnose someone as having a malignant cancer (more treatment, potentially saving lives) than falsely diagnose a cancer as benign (less treatment, potentially losing lives)

Test data = 171		Prediction	
		0	1
Actual	0	105	3
	1	1	62

Recall score = 98%

The model performs very well

Building the Model: SVM

```
# Import support vector classifier
from sklearn.svm import SVC

svc = SVC(random_state=42)

# hyperparameter values to be tuned
param_grid = {'C': np.logspace(-2,2,5),
              'gamma': [1,0.1,0.01,0.001,0.0001],
              'kernel': ['rbf','linear']}

# the tuning
svc_gridcv = GridSearchCV(estimator=svc, param_grid=param_grid, cv=5,
                                  scoring='recall',verbose=3)
svc_gridcv.fit(X_train, y_train.values.ravel())
```

Best hyperparameters:

- C = 100
- gamma = 1
- kernel = linear

Evaluating the Model: SVM

Test data = 171		Prediction	
		0	1
Actual	0	104	4
	1	2	61

Recall score = 97% → slightly lower since there is 1 more false negative

Overall, logistic regression has better performance

Dimensionality Reduction with PCA

PCA to reduce feature dimensions, and see if the results can be improved

Use the whole dataset again (minus `id` and `Unnamed: 32`) → train-test split to prevent information leakage

Scale the features → mean 0, variance 1 for all features

```python
# Import standardscaler from sklearn
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

# Scale the training and test set
X_train_std = scaler.transform(X_train)
X_test_std = scaler.transform(X_test)
```

Dimensionality Reduction with PCA

Use 5 principal components (PC) as baseline

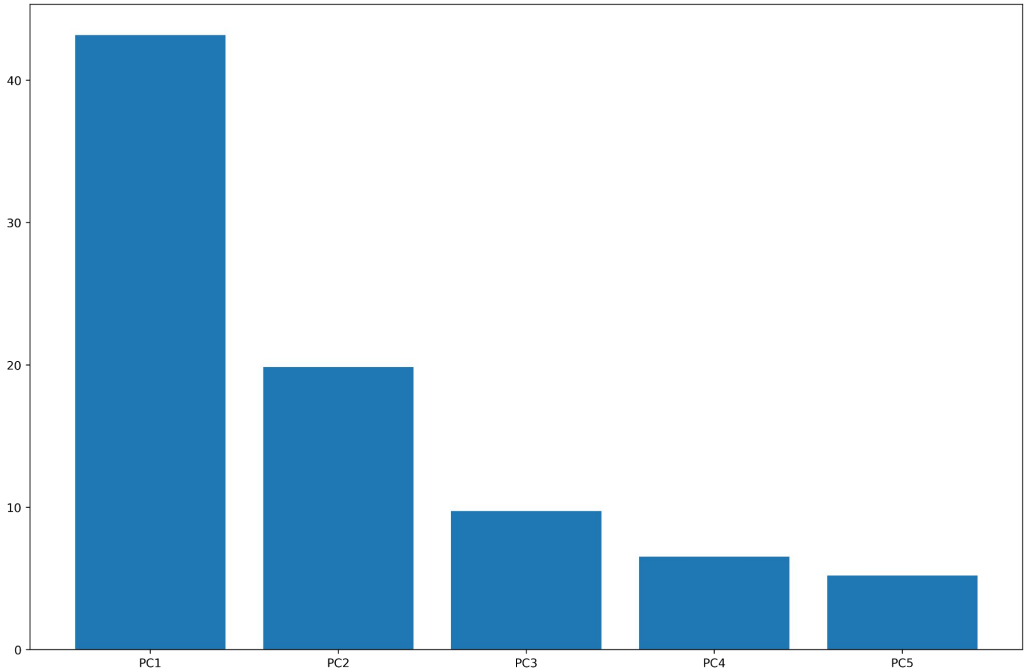
```
from sklearn.decomposition import PCA

# Let's try 5 components
pca = PCA(n_components=5)
pca.fit(X_train_std)
pcs = pca.transform(X_train_std)

pdf = pd.DataFrame(data = pcs, columns = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
pdf['diagnosis'] = y_train['diagnosis'].values
pdf.head()
```

Top 5 rows in PC coordinates

	PC1	PC2	PC3	PC4	PC5	diagnosis
0	-3.084842	-2.158704	-0.339875	-0.216630	1.545359	0
1	-2.182647	-0.617571	0.447208	-0.150818	2.860346	0
2	2.049959	2.328953	1.169407	-1.814647	-0.115599	0
3	-2.809267	-0.950791	-0.305102	-0.380816	0.547114	0
4	-1.913537	-1.072582	0.498365	1.311554	-0.204212	0

Dimensionality Reduction with PCA

Skree plot:

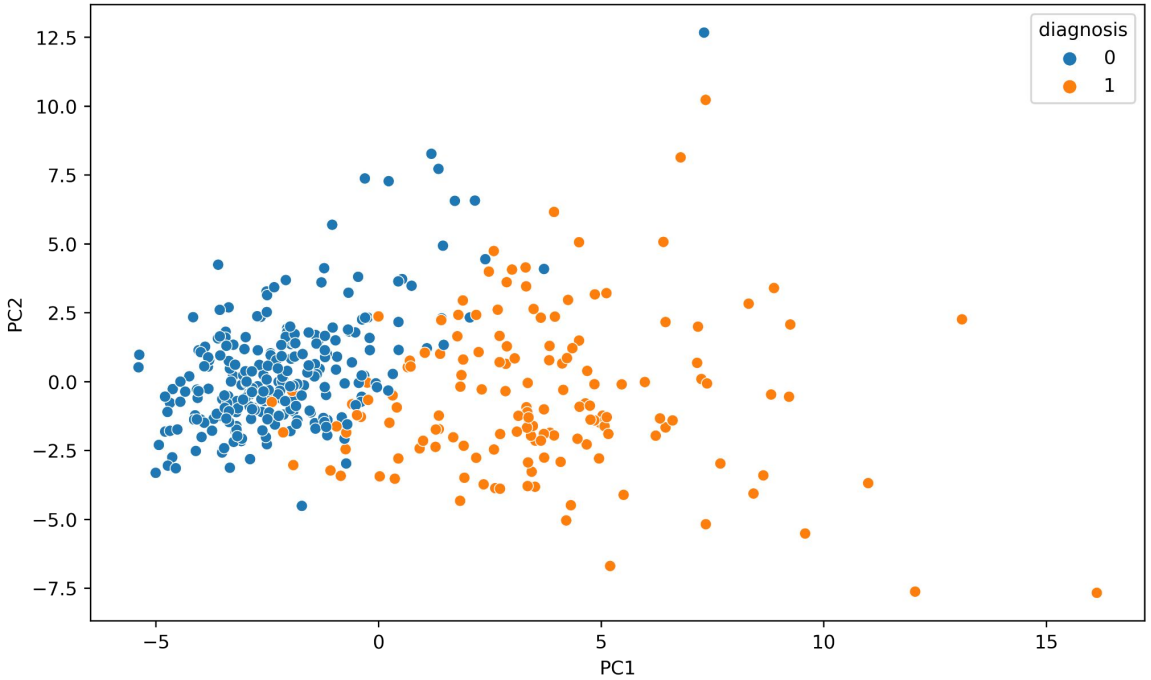


Needs at least 5 PC to preserve 80% information in the data

But is it really necessary?

Dimensionality Reduction with PCA

Plot the data in 2 PC's



The separation between the cases is very clear

Two PC's are all we need!

Building the Model: PCA + Logistic Regression

Using only 2 PC's and transform the features into PC coordinates

```
# Using 2 PC's only
pca = PCA(n_components=2)
pca.fit(X_train_std)

# Transform the train and test set
X_train_pca = pca.transform(X_train_std)
X_test_pca = pca.transform(X_test_std)
```

Building the model again with similar method (grid search)

Best hyperparameters:

- C = 0.1
- penalty = l1
- solver = liblinear

Evaluating the Model: PCA + Logistic Regression

Test data = 171		Prediction	
		0	1
Actual	0	106	2
	1	1	62

The recall score has not improved

Accuracy increases to 98% since one false positive is removed

Building the Model: PCA + SVM

```
# Create an instance
svc = SVC(random_state=42)

# hyperparameter values to be tuned
param_grid = {'C': np.logspace(-2,2,5),
              'gamma': [1,0.1,0.01,0.001,0.0001],
              'kernel': ['rbf','linear']}

# the tuning
svc_gridcv = GridSearchCV(estimator=svc, param_grid=param_grid, cv=5,
                                    scoring='recall',verbose=3)
svc_gridcv.fit(X_train_pca, y_train.values.ravel())
```

Best hyperparameters:

- C = 1
- gamma = 1
- kernel = linear

Evaluating the Model: PCA + SVM

Test data = 171		Prediction	
		0	1
Actual	0	106	2
	1	3	60

The recall score has slightly dropped to 95% (more false negatives)

Accuracy increases to 97% (more correct predictions)

Overall, logistic regression still performs better

Conclusion

1. There are more benign cases than malignant cases in this dataset → normal since severe cases should be rarer.
2. Malignant cases have larger values on all of the features.
3. The `_mean` features are highly correlated with the `_worst` features, but `_worst` have higher correlations with `diagnosis`.
4. The target `diagnosis` in this dataset is linearly separable. Therefore, logistic regression and SVM perform very well.
5. Using PCA to reduce the dimension of the data does not improve model performance significantly, but it is very time-efficient and easy to implement for this dataset.