Breast Cancer Classification

Predict whether a tumor is benign or malignant.

Identify correlations between the following 9 independent variables and the class of the tumor (benign or malignant).

- Clump Thickness
- Uniformity of Cell size
- Uniformity of cell shape
- Marginal adhesion
- Single epithelial cell
- Bare Nuclei
- Bland chromatin
- Normal nucleoli
- Mitosis

Importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing the dataset

```
df = pd.read_csv('breast_cancer.csv')
X = df.iloc[:, 1:-1].values
y = df.iloc[:, -1].values
```

```
df.head()
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	B Chrom
0	1000025	5	1	1	1	2	1	
1	1002945	5	4	4	5	7	10	
2	1015425	3	1	1	1	2	2	
3	1016277	6	8	8	1	3	4	
4	1017023	4	1	1	3	2	1	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Sample code number           683 non-null    int64
 1   Clump Thickness              683 non-null    int64
 2   Uniformity of Cell Size      683 non-null    int64
 3   Uniformity of Cell Shape     683 non-null    int64
 4   Marginal Adhesion            683 non-null    int64
 5   Single Epithelial Cell Size  683 non-null    int64
 6   Bare Nuclei                  683 non-null    int64
 7   Bland Chromatin              683 non-null    int64
 8   Normal Nucleoli              683 non-null    int64
 9   Mitoses                      683 non-null    int64
 10  Class                        683 non-null    int64
dtypes: int64(11)
memory usage: 58.8 KB
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Training the Logistic Regression model on the Training set

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[103   4]
 [  5  59]]
```

Computing the accuracy with k-Fold Cross Validation

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 96.87 %
Standard Deviation: 1.57 %
```

```
x_set,y_set=X_test,y_test
X_set, y_set = (X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict((np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('blue', 'black'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-21f5252ef4b8> in <cell line: 5>()
      3 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop =
X_set[:, 0].max() + 10, step = 0.25),
      4                      np.arange(start = X_set[:, 1].min() - 1000, stop =
X_set[:, 1].max() + 1000, step = 0.25))
----> 5 plt.contourf(X1, X2, classifier.predict((np.array([X1.ravel(),
X2.ravel()]).T)).reshape(X1.shape),
      6              alpha = 0.75, cmap = ListedColormap(('salmon',
'dodgerblue')))
      7 plt.xlim(X1.min(), X1.max())

                                  3 frames
/usr/local/lib/python3.10/dist-packages/sklearn/base.py in
_check_n_features(self, X, reset)
    387
    388         if n_features != self.n_features_in_:
--> 389             raise ValueError(
    390                 f"X has {n_features} features, but
{self.__class__.__name__} "
```

Colab paid products - Cancel contracts here

0s completed at 8:35 PM