ML-Assignment-4-Classification Problem

1.

```
1]: import pandas as pd
    from sklearn.datasets import load_breast_cancer

    # Load the dataset
    data = load_breast_cancer()

    # Convert to a DataFrame for easier manipulation
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['target'] = data.target

    # Display first few rows of the dataset
    df.head()
```

1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050

5 rows × 31 columns

2.

```
StandardScaler: Scaling the features ensures that all features contribute equally to the model. For algorit
important since they rely on distances between data points. Feature scaling helps improve model performance
```

```
2. Classification Algorithm Implementation
. Logistic Regression
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Logistic Regression model
log_reg = LogisticRegression(max_iter=10000)
log_reg.fit(X_train, y_train)
log_reg_accuracy = log_reg.score(X_test, y_test)
log_reg_accuracy
```

```
0.9824561403508771
```

3.

```
. Decision Tree Classifier
A Decision Tree classifier splits the data into subsets using a tree-like model of decisions
```

[8]:
```
from sklearn.tree import DecisionTreeClassifier

# Decision Tree model
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
dt_accuracy = dt.score(X_test, y_test)
dt_accuracy
```

[8]: 0.9415204678362573

[ ]:

4.

```
# Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
rf_accuracy = rf.score(X_test, y_test)
rf_accuracy
```

0]: 0.9707602339181286

```
. Support Vector Machine (SVM)
SVM tries to find the hyperplane that best separates the classes.
```

2]:
```
from sklearn.svm import SVC

# Support Vector Machine model
svm = SVC()
svm.fit(X_train, y_train)
svm_accuracy = svm.score(X_test, y_test)
svm_accuracy
```

2]: 0.9707602339181286

5.

14]: 0.9590643274853801

```
3. Model Comparison
Now, we compare the performance of each model based on accuracy:
```

16]:
```
# Store the accuracies of all models
accuracies = {
    'Logistic Regression': log_reg_accuracy,
    'Decision Tree': dt_accuracy,
    'Random Forest': rf_accuracy,
    'SVM': svm_accuracy,
    'k-NN': knn_accuracy
}

# Create a DataFrame for better visualization
accuracies_df = pd.DataFrame(list(accuracies.items()), columns=['Model', 'Accuracy'])

# Display the comparison
accuracies_df
```

16]:

	Model	Accuracy
0	Logistic Regression	0.982456
1	Decision Tree	0.941520
2	Random Forest	0.970760
3	SVM	0.970760