Assignment 2

CS4172 Machine Learning Lab

Name: Abhiroop Mukherjee

Enrolment Number: 510519109

Task 1

Download Cancer Wisconsin (Diagnostic) Data Set (already in the needed format). The data-set is used to recognize 2 types of cancer to be predicted (benign or malignant).

In [ ]:
```
! pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p
0\localcache\local-packages\python310\site-packages (1.4.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.
3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.21.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n
2kfra8p0\localcache\local-packages\python310\site-packages (from pandas) (1.23.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2
kfra8p0\localcache\local-packages\python310\site-packages (from pandas) (2022.2)
Requirement already satisfied: six>=1.5 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra
8p0\localcache\local-packages\python310\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [ ]:
```
import pandas as pd

FILE_PATH = "../ML_DRIVE/Assign_2/data.csv"

cancer_dataset = pd.read_csv(FILE_PATH)

cancer_dataset
```

Out[ ]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concav points_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.1471
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.0701
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.1279
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.1052
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.1043
...	...	...	...	...	...	...	...	...	...	.
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.1389
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.0979
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.0530
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.1520
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.0000

569 rows × 33 columns

In [ ]:
```
cancer_dataset.columns
```

Out[ ]:
```
Index(['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', 'Unnamed: 32'],
      dtype='object')
```

In [ ]:
```
# removing 'Unnamed: 32' and 'id'
cancer_dataset = cancer_dataset.drop('Unnamed: 32', axis=1)
cancer_dataset = cancer_dataset.drop('id', axis=1)
cancer_dataset
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmet
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	
...	...	...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	

569 rows × 31 columns

In [ ]:

```
import pandas as pd
# taking Y out of the dataframe

y = cancer_dataset[['diagnosis']]
# categorical (M or B) -> One-Hot Encode it
# y = y.join(pd.get_dummies(Y))
# y = y.drop('diagnosis', axis = 1)

# NOTE: No need to Encode it cause it happens in the LogisticRegression() and GaussianNB()
y
```

Out[ ]:

	diagnosis
0	M
1	M
2	M
3	M
4	M
...	...
564	M
565	M
566	M
567	M
568	B

569 rows × 1 columns

In [ ]:

```
# taking X out of the DataFrame
X = cancer_dataset.drop('diagnosis', axis=1)
X
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	f
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	
...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	

569 rows × 30 columns

In [ ]:

```
# Standardization
from sklearn.preprocessing import StandardScaler

X = pd.DataFrame(
    columns=X.columns,
    data=StandardScaler().fit_transform(X)
)
X
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fr
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	
...	...	...	...	...	...	...	...	...	...	
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285	2.320965	-0.312589	
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043	1.263669	-0.217664	
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588	0.105777	-0.809117	
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944	2.658866	2.137194	
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873	-1.261820	-0.820070	

569 rows × 30 columns

Task 2

Implement Logistic regression using scikit-learn package in python after splitting the dataset 80:10:10 percent (use seed = 5 for splitting).

In [ ]:
```
! pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2
kfra8p0\localcache\local-packages\python310\site-packages (1.1.2)
Requirement already satisfied: joblib>=1.0.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n
2kfra8p0\localcache\local-packages\python310\site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n
2kfra8p0\localcache\local-packages\python310\site-packages (from scikit-learn) (1.23.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2
kfra8p0\localcache\local-packages\python310\site-packages (from scikit-learn) (1.9.0)
```

In [ ]:
```python
from random import random
from sklearn.model_selection import train_test_split

X_train, X_rem, y_train, y_rem = train_test_split(
    X, y, train_size=0.8, random_state=5)
X_valid, X_test, y_valid, y_test = train_test_split(
    X_rem, y_rem, train_size=0.5, random_state=5)
```

In [ ]:
```python
print(f"len(X_train)={len(X_train)}   len(y_train)={len(y_train)}")
print(f"len(X_valid)={len(X_valid)}   len(y_train)={len(y_valid)}")
print(f"len(X_test)={len(X_test)}   len(y_train)={len(y_test)}")
```

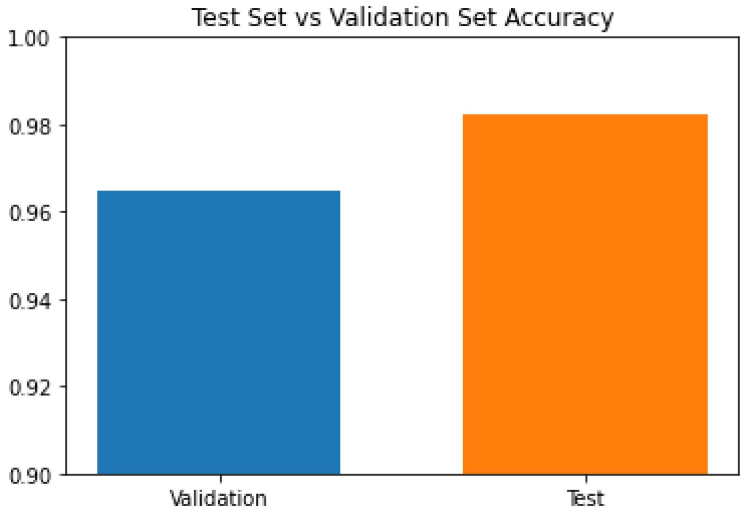
```
len(X_train)=455   len(y_train)=455
len(X_valid)=57   len(y_train)=57
len(X_test)=57   len(y_train)=57
```

In [ ]:
```python
from sklearn.linear_model import LogisticRegression

log_res_model = LogisticRegression().fit(X_train, y_train.iloc[:,0])

acc_valid = log_res_model.score(X_valid, y_valid.iloc[:, 0])
acc_test = log_res_model.score(X_test, y_test.iloc[:, 0])
print(f"Validation Set Accuracy: {acc_valid}")
print(f"Test Set Accuracy: {acc_test}")
```

```
Validation Set Accuracy: 0.9649122807017544
Test Set Accuracy: 0.9824561403508771
```

In [ ]:
```python
import matplotlib.pyplot as plt

x_axis = [0]
bar_width = 0.2
sep = 0.1
```

```
plt.bar(x_axis,[acc_valid], width=bar_width, label='Validation Set')
plt.bar([x + bar_width + sep for x in x_axis],[acc_test], width= bar_width, label='Test Set')
plt.ylim(top = 1, bottom=0.9)
plt.title('Test Set vs Validation Set Accuracy')
plt.xticks(ticks=[0.0, 0.3], labels=['Validation', 'Test'])
plt.show()
```



Helper Function

In [ ]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression


def getLogisticRegressionInfo(
        X_train: 'pd.DataFrame',
        y_train: 'pd.DataFrame',
        X_valid: 'pd.DataFrame',
        y_valid: 'pd.DataFrame',
        # saga as this solver supports all the penalty type
        solver: 'str' = 'saga',
        penalty: 'str' = 'l2',
        inv_reg_str: 'float' = 1.0,
        max_iter: 'int' = 10000) -> 'list':
    # make and train the model
```

```
    # NOTE: Note that regularization is applied by default
    model = LogisticRegression(
        solver=solver,
        penalty=penalty,
        C=inv_reg_str,
        max_iter=max_iter
    ).fit(X_train, y_train['diagnosis'])

    # find it's accuracy -> correct predict / total predict
    accuracy = model.score(X_valid, y_valid)

    return [solver, penalty, inv_reg_str, accuracy] + \
        model.coef_.tolist()[0]
```

Task 3

Use 'newton-cg', 'lbfgs', 'liblinear' solver to train the Logistic regression model, and create a table for the coefficients of all the features along with accuracy.

In [ ]:
```
import pandas as pd

newton_cg_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, solver='newton-cg'
)
lbfgs_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, solver='lbfgs'
)
liblinear_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, solver='liblinear'
)

pd.DataFrame(
    columns=['solver', 'penalty', 'inv_reg_str', 'accuracy']
    + [f"{column_name}_coef" for column_name in X_train.columns],
    data=[newton_cg_res, lbfgs_res, liblinear_res]
)
```

Out[ ]:

	solver	penalty	inv_reg_str	accuracy	radius_mean_coef	texture_mean_coef	perimeter_mean_coef	area_mean_coef	smoothness_mean_coef	compac
0	newton-cg	l2	1.0	0.964912	0.559192	0.509931	0.527163	0.573327	0.168834	
1	lbfgs	l2	1.0	0.964912	0.559265	0.509921	0.527237	0.573355	0.168756	
2	liblinear	l2	1.0	0.964912	0.541303	0.503068	0.511670	0.587558	0.158401	

3 rows × 34 columns

Task 4

Use 'l1', 'l2', 'none' penalty to train the Logistic regression model, and create a table for the coefficients of all the features along with accuracy.

In [ ]:
```
l1_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, penalty='l1'
)

l2_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, penalty='l2'
)

none_res = getLogisticRegressionInfo(
    X_train, y_train, X_valid, y_valid, penalty='none'
)

pd.DataFrame(
    columns=['solver', 'penalty', 'inv_reg_str', 'accuracy']
    + [f"{column_name}_coef" for column_name in X_train.columns],
    data=[l1_res, l2_res, none_res]
)
```

Out[ ]:

	solver	penalty	inv_reg_str	accuracy	radius_mean_coef	texture_mean_coef	perimeter_mean_coef	area_mean_coef	smoothness_mean_coef	compactr
0	saga	l1	1.0	0.982456	0.022894	0.071363	0.000000	0.000000	0.000000	
1	saga	l2	1.0	0.964912	0.558347	0.525535	0.527356	0.576819	0.167861	
2	saga	none	1.0	0.964912	0.183619	-1.604679	-0.144524	0.171642	0.766475	

3 rows × 34 columns

Task 5

Vary the l1 penalty over the range (0.1, 0.25, 0.75, 0.9) and compare the coefficients of the features.

In [ ]:

```
pd.DataFrame(
    columns=['solver', 'penalty', 'inv_reg_str', 'accuracy']
    + [f"{column_name}_coef" for column_name in X_train.columns],
    data=[
        getLogisticRegressionInfo(
            X_train, y_train, X_valid, y_valid,
            inv_reg_str=penalty
        )
        for penalty in [0.1, 0.25, 0.75, 0.9]
    ]
)
```

Out[ ]:

	solver	penalty	inv_reg_str	accuracy	radius_mean_coef	texture_mean_coef	perimeter_mean_coef	area_mean_coef	smoothness_mean_coef	compactr
0	saga	l2	0.10	0.982456	0.385834	0.418654	0.375014	0.371802	0.142495	
1	saga	l2	0.25	0.982456	0.457678	0.510483	0.441731	0.453569	0.157454	
2	saga	l2	0.75	0.964912	0.539266	0.543224	0.513195	0.554779	0.165231	
3	saga	l2	0.90	0.964912	0.551326	0.533605	0.522748	0.568590	0.166193	

4 rows × 34 columns

Task 6

Estimate the average accuracy of the Naive Bayes algorithm using 5-fold cross-validation using a scikit-learn package in python. Plot the bar graph using matplotlib

In [ ]:
```
from random import randint
from sklearn.model_selection import KFold
from sklearn.naive_bayes import BernoulliNB

data = []

for train_ids, test_ids in \
        KFold(n_splits=5, shuffle=True, random_state=randint(1, 100))\
        .split(X):
    X_train = X.iloc[train_ids]
    y_train = y.iloc[train_ids]
    X_test = X.iloc[test_ids]
    y_test = y.iloc[test_ids]
    nb_model = BernoulliNB().fit(X_train, y_train.iloc[:,0])
    accuracy = nb_model.score(X_test, y_test.iloc[:,0])
    data.append(accuracy)

kfold_df = pd.DataFrame(
    columns=['accuracy'],
    data=data
)

kfold_df
```

Out[ ]:

	accuracy
0	0.964912
1	0.921053
2	0.921053
3	0.947368
4	0.920354

In [ ]:
```
average_accuracy = kfold_df['accuracy'].mean()
print(f"average_accuracy = {average_accuracy}")
```

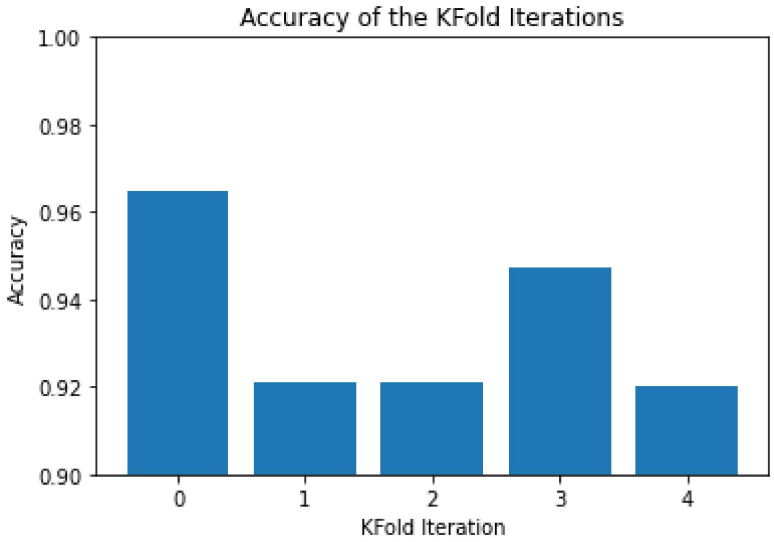
```
average_accuracy = 0.9349479894426331
```

In [ ]:
```
! pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kf
ra8p0\localcache\local-packages\python310\site-packages (3.5.3)
Requirement already satisfied: packaging>=20.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz
5n2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_q
bz5n2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (4.34.4)
Requirement already satisfied: numpy>=1.17 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2k
fra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (1.23.1)
Requirement already satisfied: cycler>=0.10 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2
kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n
2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_q
bz5n2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qb
z5n2kfra8p0\localcache\local-packages\python310\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\users\abhir\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra
8p0\localcache\local-packages\python310\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.bar(kfold_df.index,kfold_df['accuracy'])
plt.ylim(0.9, 1)
plt.xlabel('KFold Iteration')
plt.ylabel('Accuracy')
plt.title('Accuracy of the KFold Iterations')
plt.plot()
```

Out[ ]:
```
[]
```


Accuracy of the KFold Iterations
1.00
0.98
0.96
0.94
0.92
0.90
Accuracy
0
1
2
3
4
KFold Iteration