Importing the Dependencies

```
import numpy as np
import pandas as pd
import sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection & Processing

```
# loading the data from sklearn
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()

print(breast_cancer_dataset)
```

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01,
4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
```

1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
'frame': None, 'target_names': array(['malignant', 'benign'],
dtype='<U9'), 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer
wisconsin (diagnostic) dataset\
n--\n\n**Data Set
Characteristics:**\n\n   :Number of Instances: 569\n\n   :Number of
Attributes: 30 numeric, predictive attributes and the class\n\
n   :Attribute Information:\n   - radius (mean of distances from
center to points on the perimeter)\n   - texture (standard
deviation of gray-scale values)\n   - perimeter\n   - area\n
- smoothness (local variation in radius lengths)\n   -
compactness (perimeter^2 / area - 1.0)\n   - concavity (severity
of concave portions of the contour)\n   - concave points (number
of concave portions of the contour)\n   - symmetry\n   -
fractal dimension ("coastline approximation" - 1)\n\n   The mean,
standard error, and "worst" or largest (mean of the three\n
worst/largest values) of these features were computed for each image,\
n   resulting in 30 features. For instance, field 0 is Mean
Radius, field\n   10 is Radius SE, field 20 is Worst Radius.\n\n
- class:\n   - WDBC-Malignant\n   - WDBC-
Benign\n\n   :Summary Statistics:\n\n

```
================================== ====== ======\n
Min    Max\n    ================================== ====== ======\n
radius (mean):                        6.981  28.11\n    texture
(mean):                     9.71   39.28\n    perimeter (mean):
43.79  188.5\n    area (mean):                          143.5  2501.0\
n    smoothness (mean):                   0.053  0.163\n
compactness (mean):                   0.019  0.345\n    concavity
(mean):                   0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n
fractal dimension (mean):             0.05   0.097\n    radius
(standard error):              0.112  2.873\n    texture (standard
error):             0.36   4.885\n    perimeter (standard error):
0.757  21.98\n    area (standard error):                6.802  542.2\n
smoothness (standard error):          0.002  0.031\n    compactness
(standard error):         0.002  0.135\n    concavity (standard
error):           0.0    0.396\n    concave points (standard error):
0.0    0.053\n    symmetry (standard error):            0.008  0.079\n
fractal dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n
perimeter (worst):                    50.41  251.2\n    area (worst):
185.2  4254.0\n    smoothness (worst):                   0.071  0.223\
n    compactness (worst):                  0.027  1.058\n    concavity
(worst):                   0.0    1.252\n    concave points (worst):
0.0    0.291\n    symmetry (worst):                     0.156  0.664\n
fractal dimension (worst):            0.055  0.208\n
================================== ====== ======\n\n    :Missing
Attribute Values: None\n\n    :Class Distribution: 212 - Malignant,
357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street,
Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: November,
1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a
digitized image of a fine needle\naspirate (FNA) of a breast mass.
They describe\ncharacteristics of the cell nuclei present in the
image.\n\nSeparating plane described above was obtained using\
nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\
nConstruction Via Linear Programming." Proceedings of the 4th\nMidwest
Artificial Intelligence and Cognitive Science Society,\npp. 97-101,
1992], a classification method which uses linear\nprogramming to
construct a decision tree.  Relevant features\nwere selected using an
exhaustive search in the space of 1-4\nfeatures and 1-3 separating
planes.\n\nThe actual linear program used to obtain the separating
plane\nin the 3-dimensional space is that described in:\n[K. P.
Bennett and O. L. Mangasarian: "Robust Linear\nProgramming
Discrimination of Two Linearly Inseparable Sets",\nOptimization
Methods and Software 1, 1992, 23-34].\n\nThis database is also
available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd
math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n
- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature
extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993
International Symposium on \n     Electronic Imaging: Science and
```

```
Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer
diagnosis and \n     prognosis via linear programming. Operations
Research, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H.
Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques\n     to diagnose breast cancer from fine-needle aspirates.
Cancer Letters 77 (1994) \n     163-171.', 'feature_names':
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal
dimension',
       'radius error', 'texture error', 'perimeter error', 'area
error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv', 'data_module':
'sklearn.datasets.data'}
```

```
# loading the data to a data frame
data_frame = pd.DataFrame(breast_cancer_dataset.data, columns =
breast_cancer_dataset.feature_names)
```

```
# print the first 5 rows of the dataframe
data_frame.head()
```

```
   mean radius  mean texture  ...  worst symmetry  worst fractal
dimension
0        17.99         10.38  ...          0.4601
0.11890
1        20.57         17.77  ...          0.2750
0.08902
2        19.69         21.25  ...          0.3613
0.08758
3        11.42         20.38  ...          0.6638
0.17300
4        20.29         14.34  ...          0.2364
0.07678

[5 rows x 30 columns]
```

```
# adding the 'target' column to the data frame
data_frame['label'] = breast_cancer_dataset.target
```

```
# print last 5 rows of the dataframe
data_frame.tail()
```

```
     mean radius  mean texture  ...  worst fractal dimension  label
564        21.56         22.39  ...                  0.07115      0
565        20.13         28.25  ...                  0.06637      0
566        16.60         28.08  ...                  0.07820      0
567        20.60         29.33  ...                  0.12400      0
568         7.76         24.54  ...                  0.07039      1

[5 rows x 31 columns]
```

```
# number of rows and columns in the dataset
data_frame.shape
```

```
(569, 31)
```

```
# getting some information about the data
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
```

```
 29  worst fractal dimension  569 non-null    float64
 30  label                    569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
# checking for missing values
data_frame.isnull().sum()
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
label                      0
dtype: int64
```

```
# statistical measures about the data
data_frame.describe()
```

```
       mean radius  mean texture  ...  worst fractal dimension
label
count   569.000000    569.000000  ...               569.000000
569.000000
mean     14.127292     19.289649  ...                 0.083946
0.627417
std       3.524049      4.301036  ...                 0.018061
```

```
0.483918
min       6.981000      9.710000  ...                 0.055040
0.000000
25%      11.700000     16.170000  ...                 0.071460
0.000000
50%      13.370000     18.840000  ...                 0.080040
1.000000
75%      15.780000     21.800000  ...                 0.092080
1.000000
max      28.110000     39.280000  ...                 0.207500
1.000000

[8 rows x 31 columns]
```

```
# checking the distribution of Target Varibale
data_frame['label'].value_counts()
```

```
1    357
0    212
Name: label, dtype: int64
```

1 --> Benign

0 --> Malignant

```
data_frame.groupby('label').mean()
```

```
       mean radius  mean texture  ...  worst symmetry  worst fractal
dimension
label                             ...

0        17.462830     21.604906  ...        0.323468
0.091530
1        12.146524     17.914762  ...        0.270246
0.079442

[2 rows x 30 columns]
```

Separating the features and target

```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']
```

```
print(X)
```

```
     mean radius  mean texture  ...  worst symmetry  worst fractal
dimension
0          17.99         10.38  ...          0.4601
0.11890
1          20.57         17.77  ...          0.2750
0.08902
2          19.69         21.25  ...          0.3613
```

```
0.08758
3          11.42         20.38  ...          0.6638
0.17300
4          20.29         14.34  ...          0.2364
0.07678
..           ...           ...  ...             ...
...
564        21.56         22.39  ...          0.2060
0.07115
565        20.13         28.25  ...          0.2572
0.06637
566        16.60         28.08  ...          0.2218
0.07820
567        20.60         29.33  ...          0.4087
0.12400
568         7.76         24.54  ...          0.2871
0.07039

[569 rows x 30 columns]
```

```
print(Y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: label, Length: 569, dtype: int64
```

Splitting the data into training data & Testing data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

Model Training

Logistic Regression

```
model = LogisticRegression()
```

```
# training the Logistic Regression model using Training data

model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/
_logistic.py:818: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

LogisticRegression()
```

Model Evaluation

Accuracy Score

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)

print('Accuracy on training data = ', training_data_accuracy)
```

```
Accuracy on training data =  0.9494505494505494
```

```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

print('Accuracy on test data = ', test_data_accuracy)
```

```
Accuracy on test data =  0.9298245614035088
```

Building a Predictive System

```
input_data =
(13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.0576
6,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.0
023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259)

# change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array as we are predicting for one datapoint
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

```
prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The Breast cancer is Malignant')

else:
  print('The Breast Cancer is Benign')
```

```
[1]
The Breast Cancer is Benign
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  "X does not have valid feature names, but"
```