SVM_Gradient_Descent

February 27, 2024

1 Importing Necessary Libraries and Dataset

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[ ]: df = pd.read_csv("datasets/breast-cancer.csv")
```

```
[ ]: df.head()
```

```
[ ]:          id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
     0    842302         M        17.99         10.38          122.80     1001.0
     1    842517         M        20.57         17.77          132.90     1326.0
     2  84300903         M        19.69         21.25          130.00     1203.0
     3  84348301         M        11.42         20.38           77.58      386.1
     4  84358402         M        20.29         14.34          135.10     1297.0

        smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
     0          0.11840           0.27760          0.3001              0.14710
     1          0.08474           0.07864          0.0869              0.07017
     2          0.10960           0.15990          0.1974              0.12790
     3          0.14250           0.28390          0.2414              0.10520
     4          0.10030           0.13280          0.1980              0.10430

        …  radius_worst  texture_worst  perimeter_worst  area_worst  \
     0  …         25.38          17.33           184.60      2019.0
     1  …         24.99          23.41           158.80      1956.0
     2  …         23.57          25.53           152.50      1709.0
     3  …         14.91          26.50            98.87       567.7
     4  …         22.54          16.67           152.20      1575.0

        smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
     0            0.1622             0.6656           0.7119                0.2654
     1            0.1238             0.1866           0.2416                0.1860
     2            0.1444             0.4245           0.4504                0.2430
     3            0.2098             0.8663           0.6869                0.2575
     4            0.1374             0.2050           0.4000                0.1625
```

```
   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

2 Drop Unnecessary columns & Basic Exploration

```
[ ]: df = df.drop(columns=["id"])
     df.head()
```

```
[ ]:  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  …         25.38          17.33           184.60
1         0.1812  …         24.99          23.41           158.80
2         0.2069  …         23.57          25.53           152.50
3         0.2597  …         14.91          26.50            98.87
4         0.1809  …         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
```

```
2                 0.2430          0.3613                   0.08758
3                 0.2575          0.6638                   0.17300
4                 0.1625          0.2364                   0.07678

[5 rows x 31 columns]
```

[ ]:

```
df.shape
```

[ ]:

```
(569, 31)
```

[ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    float64
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
```

```
dtypes: float64(31)
memory usage: 137.9 KB
```

[ ]:
```
df.describe()
```

[ ]:
```
        diagnosis   radius_mean  texture_mean  perimeter_mean     area_mean  \
count  569.000000  5.690000e+02  5.690000e+02    5.690000e+02  5.690000e+02
mean     0.372583 -1.311195e-16 -4.027241e-16   -3.746271e-16 -1.873136e-16
std      0.483918  1.000880e+00  1.000880e+00    1.000880e+00  1.000880e+00
min      0.000000 -2.218783e+00 -2.351040e+00   -2.167411e+00 -1.727977e+00
25%      0.000000 -6.951541e-01 -7.360862e-01   -7.001605e-01 -6.931471e-01
50%      0.000000 -1.559593e-01 -6.860525e-02   -1.825548e-01 -2.041451e-01
75%      1.000000  5.188415e-01  6.213751e-01    5.730466e-01  3.921533e-01
max      1.000000  2.549701e+00  2.713815e+00    2.672459e+00  2.692856e+00

       smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
count     5.690000e+02      5.690000e+02    5.690000e+02         5.690000e+02
mean     -6.743288e-16      3.746271e-17   -7.492542e-17         1.373633e-16
std       1.000880e+00      1.000880e+00    1.000880e+00         1.000880e+00
min      -2.553720e+00     -1.797808e+00   -1.231526e+00        -1.331448e+00
25%      -7.312768e-01     -7.746323e-01   -7.761902e-01        -7.489678e-01
50%      -9.924930e-03     -1.520545e-01   -2.835775e-01        -3.706856e-01
75%       6.786382e-01      6.313915e-01    5.676350e-01         6.322354e-01
max       2.859469e+00      2.898375e+00    3.096935e+00         3.027831e+00

       symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
count   5.690000e+02  …  5.690000e+02   5.690000e+02     5.690000e+02
mean   -1.248757e-17  …  5.119904e-16   1.467289e-16     1.748260e-16
std     1.000880e+00  …  1.000880e+00   1.000880e+00     1.000880e+00
min    -2.637785e+00  … -1.905103e+00  -2.310558e+00    -1.855636e+00
25%    -7.259291e-01  … -6.770523e-01  -7.572356e-01    -6.961075e-01
50%    -1.483456e-05  … -2.032373e-01  -1.486389e-02    -2.298878e-01
75%     6.372703e-01  …  4.784760e-01   6.709298e-01     5.490950e-01
max     2.815013e+00  …  2.782280e+00   2.803748e+00     2.837185e+00

         area_worst  smoothness_worst  compactness_worst  concavity_worst  \
count  5.690000e+02      5.690000e+02       5.690000e+02     5.690000e+02
mean   2.497514e-17     -3.621395e-16       1.748260e-16     1.123881e-16
std    1.000880e+00      1.000880e+00       1.000880e+00     1.000880e+00
min   -1.523417e+00     -2.363638e+00      -1.653239e+00    -1.410523e+00
25%   -6.678659e-01     -7.072126e-01      -7.133651e-01    -7.820227e-01
50%   -2.241506e-01     -1.840221e-02      -2.062361e-01    -1.661471e-01
75%    3.463034e-01      6.282361e-01       6.018783e-01     6.358086e-01
max    3.006522e+00      2.652495e+00       3.029357e+00     2.830895e+00

       concave points_worst  symmetry_worst  fractal_dimension_worst
count          5.690000e+02    5.690000e+02             5.690000e+02
```

```
mean           2.247763e-16   -8.491548e-16              6.243785e-17
std            1.000880e+00    1.000880e+00              1.000880e+00
min           -1.745063e+00   -2.697138e+00             -1.941608e+00
25%           -7.563999e-01   -6.804106e-01             -7.380864e-01
50%           -2.234689e-01    2.570493e-03             -1.092060e-01
75%            7.125100e-01    6.189717e-01              5.915045e-01
max            2.685877e+00    2.863359e+00              2.995616e+00

[8 rows x 31 columns]
```

3 Categorise Columns

[ ]:

```python
num_cols = list(df.select_dtypes(include=['int', 'float']).columns)
cat_cols = list(df.select_dtypes(exclude=['int', 'float']).columns)
```

[ ]:

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OrdinalEncoder
from sklearn.impute import SimpleImputer
from sklearn.base import BaseEstimator, TransformerMixin
```

4 Custom IQR Removal Transformer

[ ]:

```python
class Outlier_Remover(BaseEstimator, TransformerMixin):

    def __init__(self, action='keep'):
        self.action = action

    def fit(self, X, y=None):
        self.median_ = np.median(X, axis=0)
        return self

    def transform(self, X):
        Q1 = np.percentile(X, 25, axis=0)
        Q3 = np.percentile(X, 75, axis=0)

        IQR = Q3 - Q1

        lower = Q1 - 1.5*IQR
        upper = Q3 + 1.5*IQR

        outlier_mask = (X < lower) | (X > upper)

        if self.action == 'drop':
            return X[~outlier_mask]
        else:
            for i in range(X.shape[1]):
                X[:, i][outlier_mask[:, i]] = self.median_[i]
            return X
```

5 Define Pipeline's for both columns

```
[ ]: cat_preprocessor = Pipeline(steps=[
         ("cat_null_handler",SimpleImputer(missing_values=np.
       ↪nan,strategy="most_frequent")),
         ("cat_enocder",OrdinalEncoder()),
     ])


     num_preprocessor = Pipeline(steps=[
         ("num_null_handler",SimpleImputer(missing_values=np.nan,strategy='median')),
         ("num_outlier_remover",Outlier_Remover(action="keep")),
         ("num_scaler",StandardScaler()),
     ])
```

6 Preprocess data with Pipeline

```
[ ]: df[num_cols] = num_preprocessor.fit_transform(df[num_cols])
```

```
[ ]: df[cat_cols] = cat_preprocessor.fit_transform(df[cat_cols])
```

```
[ ]: df.head()
```

```
[ ]:    diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
     0        1.0     1.335706     -2.183545        1.526900   1.477828
     1        1.0     2.168713     -0.336098        1.999151   2.692856
     2        1.0     1.884587      0.533878        1.863554   2.233014
     3        1.0    -0.785558      0.316384       -0.587475  -0.821005
     4        1.0     2.078309     -1.193573        2.102017   2.584438

        smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
     0         1.708051         -0.152054       -0.283578             2.887302
     1        -0.858619         -0.466377        0.107062             0.680991
     2         1.037027          1.359343        1.809179             2.336656
     3        -0.009925         -0.152054        2.486945             1.685632
     4         0.327875          0.750470        1.818421             1.659821

        symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
     0       2.646196  …      2.313300      -1.400167         2.761488
     1       0.084394  …      2.219021      -0.357761         1.873778
     2       1.169045  …      1.875747       0.005710         1.657012
     3      -0.000015  …     -0.217742       0.172015        -0.188255
```

```
4       0.071733  …      1.626752      -1.513323         1.646690

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0   -0.224151          1.429505          -0.206236         2.497158
1   -0.224151         -0.369836          -0.404542        -0.084360
2    2.425960          0.595436           1.460157         1.061762
3   -0.532056         -0.018402          -0.206236         2.359931
4    2.078659          0.267431          -0.260319         0.785112

   concave points_worst  symmetry_worst  fractal_dimension_worst
0              2.296076        0.002570                 2.739080
1              1.087084       -0.152067                 0.548993
2              1.955000        1.701432                 0.443446
3              2.175786        0.002570                -0.109206
4              0.729259       -0.981094                -0.348151

[5 rows x 31 columns]
```

7 Train Test Split

[ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,␣
 ↪random_state=42)
```

8 Train Model & Make predictions

[ ]:

```
from sklearn.linear_model import SGDClassifier
svm = SGDClassifier()
svm.fit(X_train,y_train)
y_pred = svm.predict(X_test)
print(y_pred)
```

```
['B' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'M' 'B' 'B'
 'B' 'M' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
 'M' 'B' 'M' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B'
 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'M'
 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'B' 'M' 'M' 'M' 'M' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'B' 'B' 'B' 'M'
 'B' 'B' 'M' 'B' 'B' 'M']
```

9 Evaluate the model

[ ]:

```
from sklearn.metrics import accuracy_score
accuracy_score(svm.predict(X_test),y_test)
```

```
[ ]: 0.9736842105263158
```

```
[ ]: from sklearn.metrics import classification_report
     print(classification_report(svm.predict(X_test),y_test))
```

```
              precision    recall  f1-score   support

           B       1.00      0.96      0.98        74
           M       0.93      1.00      0.96        40

    accuracy                           0.97       114
   macro avg       0.97      0.98      0.97       114
weighted avg       0.98      0.97      0.97       114
```

10 Conclusion:

The model is good compared to normal svm it has an accuracy of 97 and with good precision, reca
Hence, gradient descent helps in model performance.