Logistic_regression

September 30, 2020

1 Regularization with logestic regression

Regularization helps to solve over fitting problem in machine learning. Simple model will be a very poor generalization of data. At the same time, complex model may not perform well in test data due to over fitting. We need to choose the right model in between simple and complex model. Regularization helps to choose preferred model complexity, so that model is better at predicting. Regularization is nothing but adding a penalty term to the objective function and control the model complexity using that penalty term. It can be used for many machine learning algorithms.

Regularization of linear models Regularization is a method for "constraining" or "regularizing" the size of the coefficients, thus "shrinking" them towards zero. It reduces model variance and thus minimizes overfitting. If the model is too complex, it tends to reduce variance more than it increases bias, resulting in a model that is more likely to generalize.

Our aim is to locate the optimum model complexity, and thus regularization is useful when we believe our model is too complex.

1.1 Logestic regression without regularization

[30]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import seaborn as sns
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

[31]:
```
breast_cancer=pd.read_csv(r"D:\msc3\machine learning\lab7\data.csv")
breast_cancer.head(10)
```

[31]:
```
         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
5    843786         M        12.45         15.70           82.57      477.1
6    844359         M        18.25         19.98          119.60     1040.0
7  84458202         M        13.71         20.83           90.20      577.9
8    844981         M        13.00         21.82           87.50      519.8
9  84501001         M        12.46         24.04           83.97      475.9

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760         0.30010              0.14710
1          0.08474           0.07864         0.08690              0.07017
2          0.10960           0.15990         0.19740              0.12790
3          0.14250           0.28390         0.24140              0.10520
4          0.10030           0.13280         0.19800              0.10430
5          0.12780           0.17000         0.15780              0.08089
6          0.09463           0.10900         0.11270              0.07400
7          0.11890           0.16450         0.09366              0.05985
8          0.12730           0.19320         0.18590              0.09353
9          0.11860           0.23960         0.22730              0.08543

   …  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
5  …         15.47          23.75           103.40       741.6
6  …         22.88          27.66           153.20      1606.0
7  …         17.06          28.14           110.60       897.0
8  …         15.49          30.73           106.20       739.3
9  …         15.09          40.68            97.65       711.4

   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
5            0.1791             0.5249           0.5355                0.1741
6            0.1442             0.2576           0.3784                0.1932
7            0.1654             0.3682           0.2678                0.1556
8            0.1703             0.5401           0.5390                0.2060
9            0.1853             1.0580           1.1050                0.2210

   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          0.3985                   0.12440
6          0.3063                   0.08368
7          0.3196                   0.11510
8          0.4378                   0.10720
9          0.4366                   0.20750

[10 rows x 32 columns]
```

[32]:
```python
print("Number of data:"+str(len(breast_cancer.index)))
```

```
Number of data:569
```

[33]:
```python
breast_cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
```

```
 26   smoothness_worst         569 non-null    float64
 27   compactness_worst        569 non-null    float64
 28   concavity_worst          569 non-null    float64
 29   concave points_worst     569 non-null    float64
 30   symmetry_worst           569 non-null    float64
 31   fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

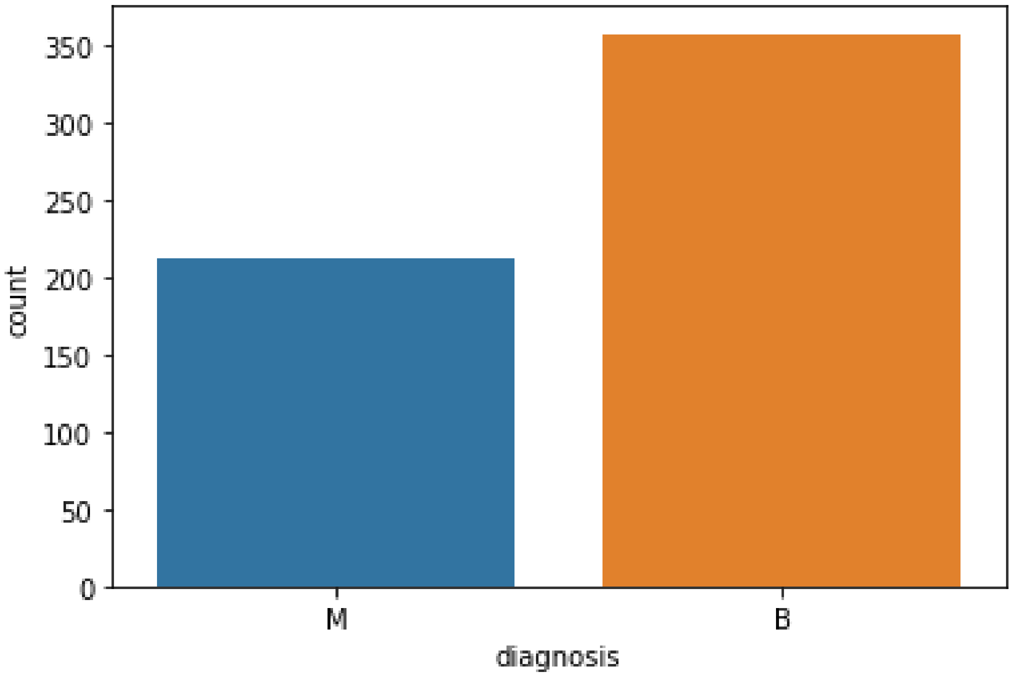
[34]:
```
breast_cancer.isnull().any()
```

[34]:
```
id                         False
diagnosis                  False
radius_mean                False
texture_mean               False
perimeter_mean             False
area_mean                  False
smoothness_mean            False
compactness_mean           False
concavity_mean             False
concave points_mean        False
symmetry_mean              False
fractal_dimension_mean     False
radius_se                  False
texture_se                 False
perimeter_se               False
area_se                    False
smoothness_se              False
compactness_se             False
concavity_se               False
concave points_se          False
symmetry_se                False
fractal_dimension_se       False
radius_worst               False
texture_worst              False
perimeter_worst            False
area_worst                 False
smoothness_worst           False
compactness_worst          False
concavity_worst            False
concave points_worst       False
symmetry_worst             False
fractal_dimension_worst    False
dtype: bool
```

2 Analyzing data

[35]:
```
sns.countplot("diagnosis",data=breast_cancer)
```

[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x267a0ae5dc8>
```



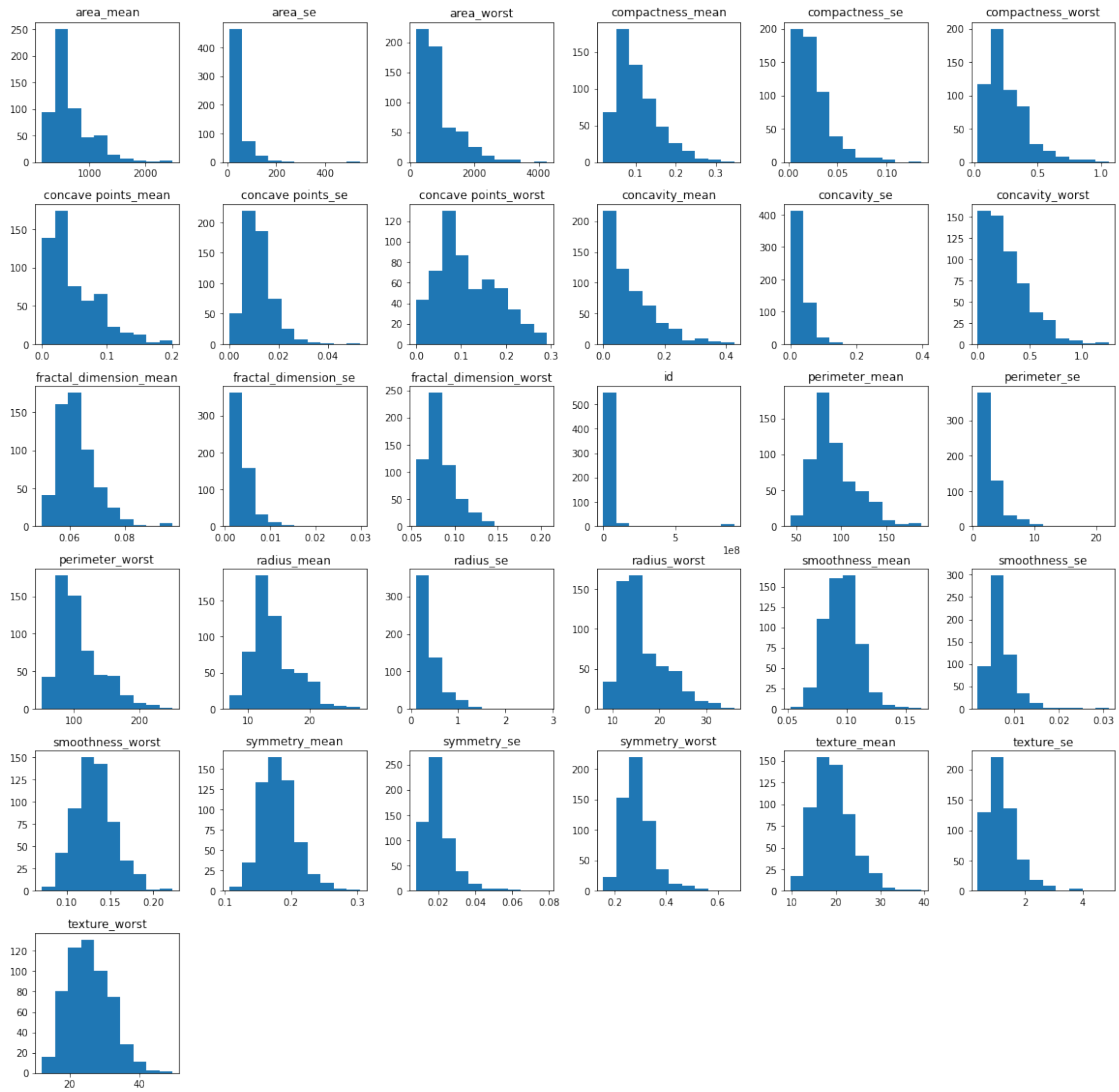
[36]:
```
breast_cancer.diagnosis.value_counts()
```

[36]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

[37]:
```
breast_cancer.hist(bins=10,figsize=(20,20),grid=False)
```

[37]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0BFC208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0C28148>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0C5A5C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0C920C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0CBEB88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0CF8688>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0D31188>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0D5CC88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0D6D508>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0DA7148>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0E02D08>,
```

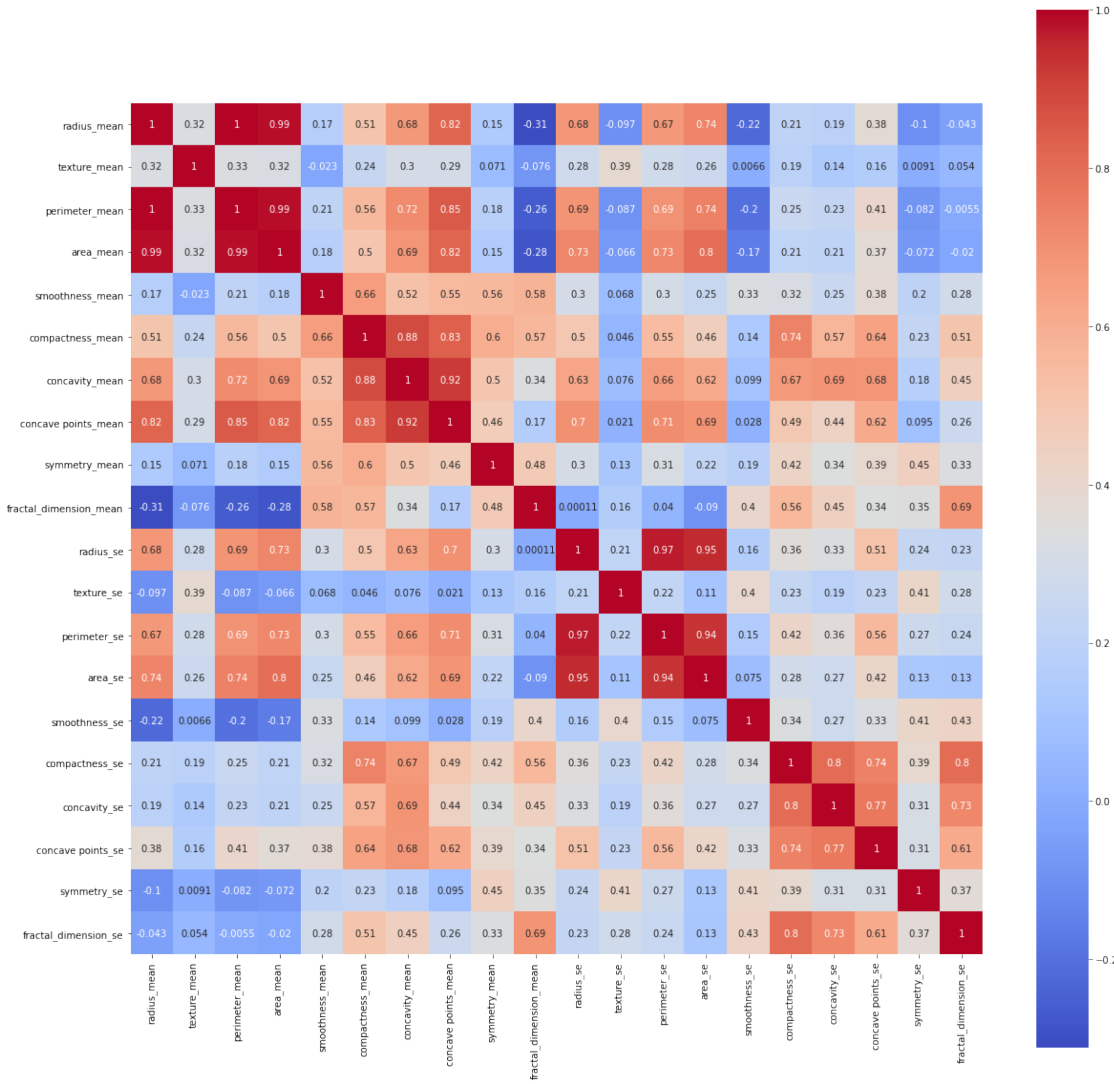
```
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0E3CAC8>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0E747C8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0EAD548>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0EE5308>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0F1E088>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A1642DC8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A167AB08>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0FF1308>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0ABA688>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0797048>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0AEC148>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A1064408>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0776F48>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0AD7DC8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A169DC88>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0826B08>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A086BF88>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A089B808>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A08D4688>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x00000267A090E548>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A09463C8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0982308>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A09BA248>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A09F4188>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x00000267A0A2E108>]],
dtype=object)
```



[38]:
```python
X=breast_cancer.drop("diagnosis",axis=1)
y=breast_cancer["diagnosis"]
```

[39]:
```python
features_mean= list(breast_cancer.columns[2:22])
```

[40]:
```python
plt.figure(figsize=(20,20))
sns.heatmap(breast_cancer[features_mean].corr(), annot=True, square=True, cmap='coolwarm')
plt.show()
```



[41]:
```python
breast_cancer.drop("id", axis=1, inplace=True)
```

[42]:
```python
breast_cancer.head(10)
```

[42]:
```
  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
5         M        12.45         15.70           82.57      477.1
6         M        18.25         19.98          119.60     1040.0
7         M        13.71         20.83           90.20      577.9
8         M        13.00         21.82           87.50      519.8
```

9	M	12.46	24.04	83.97	475.9

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.30010	0.14710	
1	0.08474	0.07864	0.08690	0.07017	
2	0.10960	0.15990	0.19740	0.12790	
3	0.14250	0.28390	0.24140	0.10520	
4	0.10030	0.13280	0.19800	0.10430	
5	0.12780	0.17000	0.15780	0.08089	
6	0.09463	0.10900	0.11270	0.07400	
7	0.11890	0.16450	0.09366	0.05985	
8	0.12730	0.19320	0.18590	0.09353	
9	0.11860	0.23960	0.22730	0.08543	

	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	
5	0.2087	…	15.47	23.75	103.40	
6	0.1794	…	22.88	27.66	153.20	
7	0.2196	…	17.06	28.14	110.60	
8	0.2350	…	15.49	30.73	106.20	
9	0.2030	…	15.09	40.68	97.65	

	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	
5	741.6	0.1791	0.5249	0.5355	
6	1606.0	0.1442	0.2576	0.3784	
7	897.0	0.1654	0.3682	0.2678	
8	739.3	0.1703	0.5401	0.5390	
9	711.4	0.1853	1.0580	1.1050	

	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	0.1741	0.3985	0.12440
6	0.1932	0.3063	0.08368
7	0.1556	0.3196	0.11510

```
8                0.2060         0.4378                   0.10720
9                0.2210         0.4366                   0.20750

[10 rows x 31 columns]
```

[43]:
```
diagno=pd.get_dummies(breast_cancer['diagnosis'], drop_first=True)
diagno.head()
```

[43]:
```
   M
0  1
1  1
2  1
3  1
4  1
```

[44]:
```
breast_cancer=pd.concat((breast_cancer,diagno),axis=1)
```

[45]:
```
breast_cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   diagnosis               569 non-null    object
 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
 31  M                        569 non-null    uint8
dtypes: float64(30), object(1), uint8(1)
memory usage: 138.5+ KB
```

[46]:
```
breast_cancer.drop("diagnosis", axis=1, inplace=True)
```

[47]:
```
breast_cancer.head()
```

[47]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          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  M
0          0.4601                  0.11890  1
1          0.2750                  0.08902  1
```

```
2          0.3613                  0.08758  1
3          0.6638                  0.17300  1
4          0.2364                  0.07678  1

[5 rows x 31 columns]
```

[48]:
```
X=breast_cancer.drop("M",axis=1)
y=breast_cancer["M"]
```

[49]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,␣
 ↪random_state=2)
```

[50]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e5)
logreg.fit(X_train,y_train)
predictions=logreg.predict(X_test)
```

```
C:\Users\blr0a\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:764: 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)
```

[51]:
```
from sklearn.metrics import classification_report
classification_report(y_test,predictions)
```

[51]:
```
'              precision    recall  f1-score   support\n\n           0
0.92      0.94      0.93       104\n           1       0.91      0.88      0.89
67\n\n    accuracy                           0.92       171\n   macro avg
0.92      0.91      0.91       171\nweighted avg       0.92      0.92      0.92
171\n'
```

[52]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,predictions)
```

[52]:
```
array([[98,  6],
       [ 8, 59]], dtype=int64)
```

[53]:
```
from sklearn.metrics import accuracy_score
print("train accuracy:")
```

```
print(format(logreg.score(X_train,y_train)*100.0))
print("test accuracy:")
accuracy_score(y_test,predictions)
```

```
train accuracy:
94.72361809045226
test accuracy:
```

[53]: 0.9181286549707602

[54]:
```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['features','effect_score']  #naming the dataframe␣
↪columns
print(featureScores.nlargest(10,'effect_score'))  #print 10 best features
```

```
            features    effect_score
23        area_worst   112598.431564
3          area_mean    53991.655924
13           area_se     8758.504705
22   perimeter_worst     3665.035416
2     perimeter_mean     2011.102864
20      radius_worst      491.689157
0        radius_mean      266.104917
12      perimeter_se      250.571896
21     texture_worst      174.449400
1       texture_mean       93.897508
```

[82]:
```
from sklearn.model_selection import cross_val_score
mse=cross_val_score␣
↪(logreg,X_test,y_test,scoring='neg_mean_squared_error',cv=10)
mean_mse=np.mean(mse)
print(mean_mse)
```

```
-0.08202614379084967
```

[58]:
```
from sklearn.metrics import r2_score
from sklearn import metrics
from sklearn.linear_model import Ridge
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import Ridge
```

```
ridgereg = Ridge(alpha=0, normalize=True)
ridgereg.fit(X_train, y_train)
y_pred = ridgereg.predict(X_test)
print("R-Square Value",r2_score(y_test,y_pred))
print("\n")
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_pred))
print("\n")
print ("mean_squared_error : ",metrics.mean_squared_error(y_test, y_pred))
print("\n")
print ("root_mean_squared_error : ",np.sqrt(metrics.mean_squared_error(y_test,␣
↪y_pred)))
```

```
R-Square Value 0.7183226717497466


mean_absolute_error : 0.1963291040465723


mean_squared_error :  0.06712245214759298


root_mean_squared_error :  0.25908001109231293
```

2.1 Ridge Regularization

[83]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
ridge=Ridge()
parameters={'alpha':
↪[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge_regression=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regression.fit(X_train, y_train)
```

[83]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
                                   20, 30, 35, 40, 45, 50, 55, 100]},
             scoring='neg_mean_squared_error')
```

[84]:
```
print(ridge_regression.best_params_)
print(ridge_regression.best_score_)
```

```
{'alpha': 0.01}
-0.05979886379489485
```

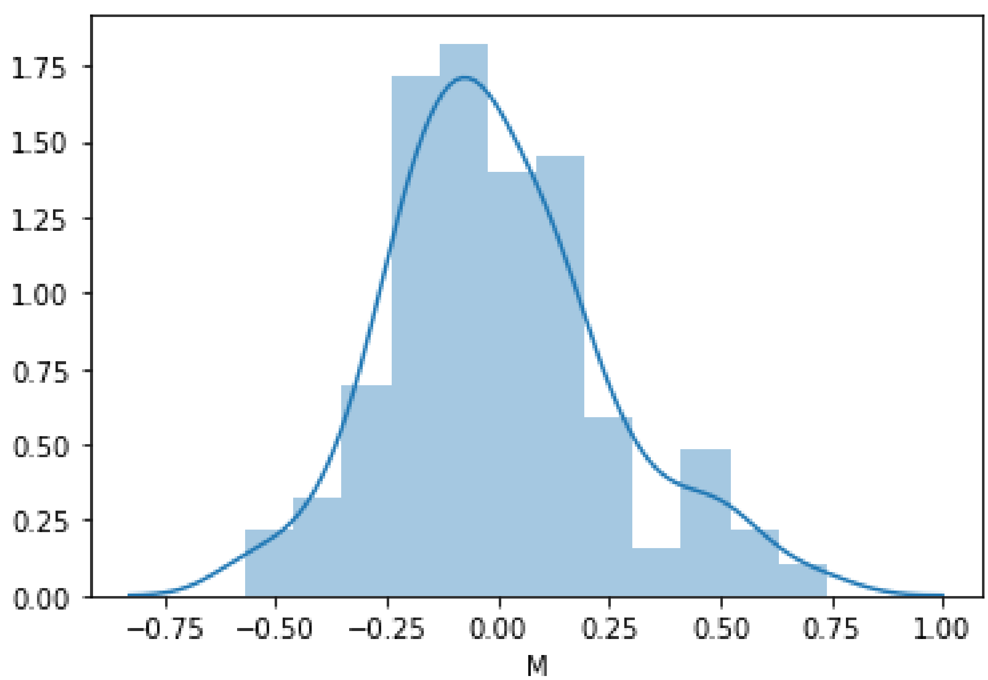
[85]:
```
ridgereg = Ridge(0.01, normalize=True)
ridgereg.fit(X_train, y_train)
ridge_pred = ridgereg.predict(X_test)
```

```
print("R-Square Value",r2_score(y_test,ridge_pred))
print("\n")
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_pred))
print("\n")
print ("mean_squared_error : ",metrics.mean_squared_error(y_test, y_pred))
print("\n")
print ("root_mean_squared_error : ",np.sqrt(metrics.mean_squared_error(y_test,␣
↪y_pred)))
```

```
R-Square Value 0.7385366367368109


mean_absolute_error : 0.19521123955034342


mean_squared_error :  0.06517942196335559


root_mean_squared_error :  0.25530260861055765
```

[86]:
```
sns.distplot(y_test-ridge_pred)
```

[86]: <matplotlib.axes._subplots.AxesSubplot at 0x267a7a8b5c8>



2.2 Lasso Regularization

[87]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
lasso=Lasso()
parameters={'alpha':
 ↪[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
lasso_regression=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)
lasso_regression.fit(X_train, y_train)
print(lasso_regression.best_params_)
print(lasso_regression.best_score_)
```

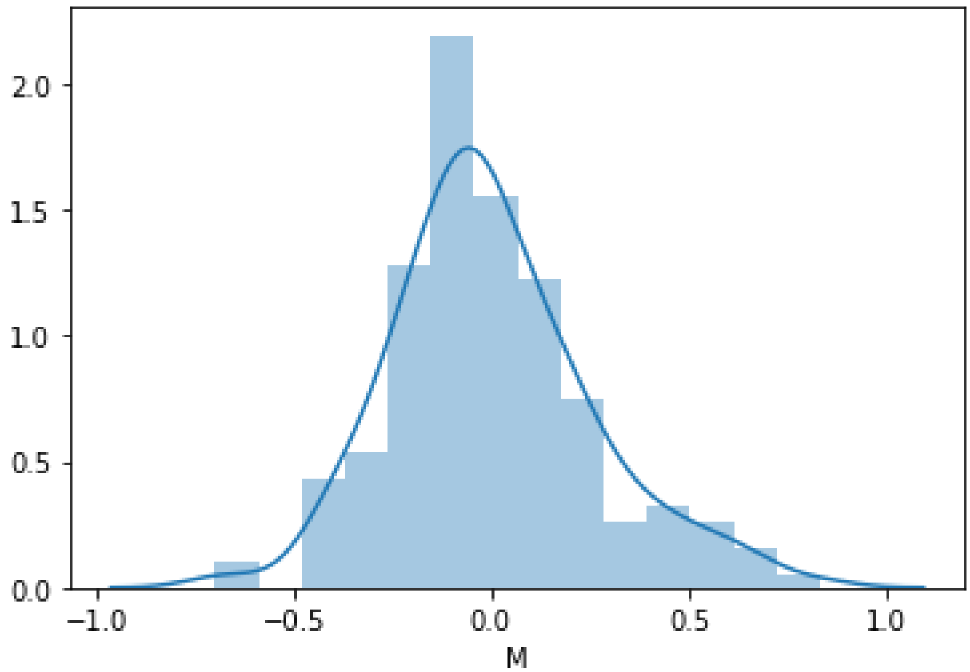
```
{'alpha': 1e-08}
-0.0623197426776443555
```

[88]:
```
lassoreg = Lasso(1e-08, normalize=True)
lassoreg.fit(X_train, y_train)
lasso_pred = lassoreg.predict(X_test)
print("R-Square Value",r2_score(y_test,lasso_pred))
print("\n")
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_pred))
print("\n")
print ("mean_squared_error : ",metrics.mean_squared_error(y_test, y_pred))
print("\n")
print ("root_mean_squared_error : ",np.sqrt(metrics.mean_squared_error(y_test,␣
 ↪y_pred)))
```

```
R-Square Value 0.7264765388015957


mean_absolute_error : 0.19521123955034342


mean_squared_error :  0.06517942196335559


root_mean_squared_error :  0.25530260861055765
```

[89]:
```
sns.distplot(y_test-lasso_pred)
```

[89]: <matplotlib.axes._subplots.AxesSubplot at 0x267a7b155c8>


2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
M

```
[ ]:
```

```
[ ]:
```