8B

March 11, 2022

[1]:
```
import numpy as np
import pandas as pd
import plotly
import plotly.figure_factory as ff
import plotly.graph_objs as go
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

[2]:
```
data = pd.read_csv('task_b.csv')
data=data.iloc[:,1:]
```

[6]:
```
X=data[['f1','f2','f3']].values
Y=data['y'].values
data.head()
```

[6]:

	f1	f2	f3	y
0	-195.871045	-14843.084171	5.532140	1.0
1	-1217.183964	-4068.124621	4.416082	1.0
2	9.138451	4413.412028	0.425317	0.0
3	363.824242	15474.760647	1.094119	0.0
4	-768.812047	-7963.932192	1.870536	0.0

[5]:
```
data.corr()['y']
```

[5]:
```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

SGD without standarscaling

[24]:
```
clf = linear_model.SGDClassifier(loss='log')
clf.fit(X,Y)
```

```
print("weight of the feature sgd with log_loss",clf.coef_)
clf = linear_model.SGDClassifier(loss='hinge')
clf.fit(X,Y)
print("weight of the feature sgd with hinge-loss",clf.coef_)
```

```
weight of the feature sgd with log_loss [[-5810.35602682 -7568.12107298
11196.01895478]]
weight of the feature sgd with hinge-loss [[13548.37300582 19610.41444714
9910.89116627]]
```

0.0.1 1. According to sgdclassifier with log_loss,The feature importance is given by weight the respective features and ranking are :-var(F3) » var(F1) » Var(F2)

0.0.2 2. According to sgdclassifier with hinge_loss, The feature importance is given by :- var(F1) » var(F3) » Var(F2)

0.0.3 3. Each time we run the model there is lot change in weights that means there is more variance (std-dev) in data which leads into change in features importance each time.

0.0.4 4. Because of variance(std-dev) in data which result into the high weight in feature but there are not important to find the feature importances.

SGD with standarscaling

[64]:
```
Sx=MinMaxScaler()
sx=StandardScaler()

scale1=sx.fit_transform(X)
scale=Sx.fit_transform(X)

clf = linear_model.SGDClassifier(loss='log')

clf.fit(scale,Y)
print(clf.coef_,"weight of the features 'by MinMaxScaler' and 'logloss' ")
clf.fit(scale1,Y)
print(clf.coef_,"weight of the features 'by StandarScaler' and 'logloss' ")
print("*"*100)




clf = linear_model.SGDClassifier(loss='hinge')
clf.fit(scale,Y)
print(clf.coef_,"weight of the features 'by MinMaxScaler' and 'hinge-loss'")
clf.fit(scale1,Y)
print(clf.coef_,"weight of the features 'by StandarScaler' and 'hinge-loss'")
```

```
[[-2.60927123  0.02583157 21.94870556]] weight of the features 'by MinMaxScaler'
and 'logloss'
```

```
[[-2.39090927 -0.31120328 11.27580304]] weight of the features 'by
StandarScaler' and 'logloss'
*******************************************************************************
********************
[[-4.70015974  1.35097432 36.41125463]] weight of the features 'by MinMaxScaler'
and 'hinge-loss'
[[-5.65847531 -3.92804409 27.30465125]] weight of the features 'by
StandarScaler' and 'hinge-loss'
```

0.0.5 1. After StandarScaling the model's output is more consistence.

0.0.6 2. After standardization there is no more effect of varience and weight of feature shows proper importances.

0.0.7 3. SGD classifier with log__loss and hinge__loss after standarscaling feature importance change :- var(F3) » var(F2) » var(F1),Weight of feature 1 and 2 reduce more compare to 3 and feature 3 is most importance feature.

0.0.8 4. Feature weight is proportional to the correlation of the feature.

[ ]: