In [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.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [2]:

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

In [3]:

```
data.head()
```

Out[3]:

	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

In [4]:

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

Out[4]:

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

In [5]:

```
data.std()
```

Out[5]:

```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

In [6]:

```
X=data[['f1','f2','f3']].values
Y=data['y'].values
print(X.shape)
print(Y.shape)
```

```
(200, 3)
(200,)
```

What if our features are with different variance

```
  * As part of this task you will observe how linear models work in case of data having feautres with different variance
  * from the output of the above cells you can observe that var(F2)>>var(F1)>>Var(F3)

  > Task1:
      1. Apply Logistic regression(SGDClassifier with logloss) on 'data' and check the feature importance
      2. Apply SVM(SGDClassifier with hinge) on 'data' and check the feature importance

  > Task2:
      1. Apply Logistic regression(SGDClassifier with logloss) on 'data' after standardization
         i.e standardization(data, column wise): (column-mean(column))/std(column) and check the feature importance
      2. Apply SVM(SGDClassifier with hinge) on 'data' after standardization
         i.e standardization(data, column wise): (column-mean(column))/std(column) and check the feature importance
```

Make sure you write the observations for each task, why a particular feautre got more importance than others

In [7]:

```
#Task -1 1
from sklearn.linear_model import SGDClassifier

clf=SGDClassifier(loss='log')
clf.fit(X,Y)
importance=clf.coef_[0]

print(f'The weight for the features are {importance}')
```

```
The weight for the features are [-2148.93447288 19892.80818586 10509.13532407]
```

In [8]:

```
#Task-1    2
from sklearn.linear_model import SGDClassifier
clf=SGDClassifier(loss='hinge')
clf.fit(X,Y)
importance=clf.coef_[0]
print(f'The weight for the features are {importance}')
```

```
The weight for the features are [ -735.64174108 34157.76031609 10415.09563052]
```

In [9]:

```
#task -2
#Column standarization
for i in range(3):
    X[:,i]=(X[:,i]-np.mean(X[:,i]))/np.std(X[:,i])
```

In [10]:

```
clf=SGDClassifier(loss='log')
clf.fit(X,Y)
importance=clf.coef_[0]

print(f'The weight for the features are {importance}')
```

```
The weight for the features are [-1.51232356e+00 -7.35482719e-03  9.52243545e+00]
```

In [11]:

```
clf=SGDClassifier(loss='hinge')
clf.fit(X,Y)
importance=clf.coef_[0]
print(f'The weight for the features are {importance}')
```

```
The weight for the features are [ 3.78322986  5.23493797 27.96607593]
```

Summary

1. If two features are highly correlated that means we can select one feature among the two.

2. Based on variance we can't say whether a feature is important or not.

3. In case 1, we found that f2 has more weight than the other two features.

4.After Standarization the SVM model weight has changed where as in logistic regression the feature importances are not change.

5. Before standarization using SVM we found that f2 has more weight but after standization we found that f3 has more weight, this is because SVM will depends on the actual values

In [ ]:

```

```

In [ ]:

```

```