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]:

```
data1=data.std()
```

In [7]:

```
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

In [8]:

```
data_y = data['y']
data_x = data.drop('y',axis = 1)
```

In [9]:

```
from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(loss = 'hinge')
clf.fit(data_x,data_y)
print(abs(clf.coef_))

clf = SGDClassifier(loss = 'log')
clf.fit(data_x,data_y)
print(abs(clf.coef_))
```

```
[[27454.23967239 29546.40758139  9985.22929229]]
[[ 7357.93895888 16047.03577689 10175.89901152]]
```

In [10]:

```
mean = data_x['f1'].mean()
std = data_x['f1'].std()

data_x['f1']-=mean
data_x['f1']/=std

mean = data_x['f2'].mean()
std = data_x['f2'].std()

data_x['f2']-=mean
data_x['f2']/=std

mean = data_x['f3'].mean()
std = data_x['f3'].std()

data_x['f3']-=mean
```

```
data_x['f3']/=std

clf = SGDClassifier(loss = 'hinge')
clf.fit(data_x,data_y)
print(abs(clf.coef_))

clf = SGDClassifier(loss = 'log')
clf.fit(data_x,data_y)
print(abs(clf.coef_))
```

```
[[ 1.48618218  0.55714659 13.42840668]]
[[ 2.67332811  0.08815571 17.20058088]]
```

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

OBSERVATIONS Since "hinge" loss and "log" loss are both loss function the order of the feature importance should be the same. When the features are not standardized, based on the loss function properties the priorites differ. For hinge loss the weight of features in order is F2>F1>F3 For log loss its F2>F3>F1 But when we properly standardize the data we can clearly get to know which features are really important to do the classification task since both of them show the same order of weights F3>F1>F2