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]:

```
from sklearn.linear_model import SGDClassifier
clfLog = SGDClassifier(loss='log')
clfLog.fit(X,Y)
print(clfLog.coef_, clfLog.intercept_)
```

```
[[14586.2516274   1462.69099437 10722.77397258]] [-129.52735674]
```

In [8]:

```
clfHinge = SGDClassifier(loss='hinge')
clfHinge.fit(X,Y)
print(clfHinge.coef_, clfHinge.intercept_)
```

```
[[   545.7447711  -37623.62519078  10060.31259476]] [66.80363062]
```

Observations:

- When data is not scaled or standardized, the feature importance seems to be random and independent of standard deviation (variance) within the data.

In [9]:

```
scaler = StandardScaler()
data_scaled = pd.DataFrame(scaler.fit_transform(data),columns=['f1','f2','f3','y'])
data_scaled.std()
```

Out[9]:

```
f1    1.002509
f2    1.002509
f3    1.002509
y     1.002509
dtype: float64
```

In [10]:

```
X_scaled=data_scaled[['f1','f2','f3']].values
Y_scaled=data_scaled['y'].values
clfLog.fit(X_scaled,Y_scaled)
clfHinge.fit(X_scaled,Y_scaled)
print(clfLog.coef_, clfLog.intercept_)
print(clfHinge.coef_, clfHinge.intercept_)
```

```
[[-4.94690128 -4.12954084 16.35655128]] [-3.40716057]
[[-2.71723638  0.70845152 24.66128722]] [-1.28762251]
```

Observations:

- When data is scaled or standardized, the feature importance is inversely dependent on standard deviation (variance) within the data. This is desired as features which have outliers are expected to have more variance and hence should be given less importance.