In [40]:
```
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

Task-1

In [17]:
```
from sklearn.linear_model import SGDClassifier
logloss = SGDClassifier(loss= 'log')
logloss.fit(X,Y)
print(logloss.coef_,logloss.intercept_)
```

```
[[13650.74928085 -1852.42280192 10130.20430125]] [8.92141119]
```

In [29]:
```
from sklearn.linear_model import SGDClassifier
hingeloss = SGDClassifier(loss= 'hinge')
hingeloss.fit(X,Y)
print(hingeloss.coef_,hingeloss.intercept_)
```

```
[[ 8807.86824203 -6483.81087347  9788.74610561]] [56.2610501]
```

Observation

In this task we had fit the model without column standardization the feature importance seems to be random and independent of standard deviation (variance) within the data.

Task - 2

In [36]:
```
scaler = StandardScaler()
x_scaled = pd.DataFrame(scaler.fit_transform(X),columns =['f1','f2','f3'])
print(x_scaled.std())
reshape = Y.reshape(-1,1)
y_scaled = pd.DataFrame(scaler.fit_transform(reshape),columns =['y'])
print(y_scaled.std())
```

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

In [45]:
```
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import SGDClassifier
logloss = SGDClassifier(loss= 'log')
logloss.fit(x_scaled,y_scaled)
print(logloss.coef_,logloss.intercept_)
```

```
[[1.20753391 1.58503085 8.79424288]] [0.46448202]
```

In [49]:
```
from sklearn.linear_model import SGDClassifier
hingeloss = SGDClassifier(loss= 'hinge')
hingeloss.fit(x_scaled,y_scaled)
print(hingeloss.coef_,hingeloss.intercept_)
```

```
[[-1.87574082 -2.05965183 15.98624399]] [-0.79522618]
```

Observation

In this task the data is 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.