In [3]:
```
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
from sklearn.linear_model import SGDClassifier
init_notebook_mode(connected=True)
```

In [4]:
```
data = pd.read_csv('task_b.csv')
data=data.iloc[:,1:]
```

In [5]:
```
data.head()
```

Out[5]:

	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 [6]:
```
data.corr()['y']
```

Out[6]:
```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

In [7]:
```
data.std()
```

Out[7]:
```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

In [8]:
```
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

Logistic Regression with SGD

In [9]:
```
clf = SGDClassifier(loss='log')
clf.fit(X,Y)

for i , j in enumerate(clf.coef_.flatten()):
    print ("f",i+1,"->",j)
```

```
f 1 -> -603.68310064344
f 2 -> -10074.531538628244
f 3 -> 9542.846409398144
```

In [10]:
```
clf = SGDClassifier(loss='hinge')
clf.fit(X,Y)

for i , j in enumerate(clf.coef_.flatten()):
    print ("f",i+1,"->",j)
```

```
f 1 -> 5051.871474115397
f 2 -> 2871.996018403727
f 3 -> 11042.581496781871
```

Observations :

1. Huge values make comparison a bit difficult.
2. In both the loss types , F3 is the most important feature and F2 is the least important feature.

In [11]:
```
scaler = StandardScaler()
X_new = scaler.fit_transform(X) #scaling X
```

In [12]:
```
clf = SGDClassifier(loss='log')
clf.fit(X_new,Y)

for i , j in enumerate(clf.coef_.flatten()):
    print ("f",i+1,"->",j)
```

```
f 1 -> 1.1876059639101975
f 2 -> -0.31355531198778314
f 3 -> 10.174481247117702
```

In [13]:

```
clf = SGDClassifier(loss='hinge')
clf.fit(X_new,Y)

for i , j in enumerate(clf.coef_.flatten()):
    print ("f",i+1,"->",j)
```

```
f 1 -> 1.0934023325145013
f 2 -> 0.4998451731163542
f 3 -> 13.009098789585808
```

Observations:

1. Since the feature importance values are in tens it is much more readable and comparable.
2. Since all the features are scaled and now on the same scale where deviation is 1 and mean is 0 , the comparision in feature importance becomes easier and efficient.
3. In this case also, for both loss , F3 is of highest importance and F2 is of lowest importance.