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 import linear_model
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)
X[0]
```

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

Out[6]:
```
array([-1.95871045e+02, -1.48430842e+04,  5.53214037e+00])
```

In [7]:
```
data1 = data.drop('y', axis = 1)
data1.columns
```

Out[7]:
```
Index(['f1', 'f2', 'f3'], dtype='object')
```

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: Fitting the model Before standadization

In [8]:
```
model_LR = linear_model.SGDClassifier(loss = 'log')
```

In [9]:
```
model_LR.fit(X, Y)
```

Out[9]:
```
SGDClassifier(loss='log')
```

In [10]:
```
imp_LR = model_LR.coef_
```

In [12]:
```
print(imp_LR)
```

```
[[5921.87568274 -871.84119588 7885.6477321 ]]
```

In [13]:
```
model_SVM = linear_model.SGDClassifier()
```

In [14]:
```
model_SVM.fit(X, Y)
```

Out[14]:
```
SGDClassifier()
```

In [15]:
```
imp_SVM = model_SVM.coef_
```

In [16]:
```
print(imp_SVM)
```

```
[[ 7105.85630403 -4545.47362328 10301.09462378]]
```

Task 2: Fitting the model After Standadization

In [17]:
```
names = ['f1', 'f2', 'f3']
```

In [18]:
```
for x in names:
  data[x] = (data[x] - np.mean(data[x])) / data[x].std()
```

In [19]:
```
model_LR = linear_model.SGDClassifier(loss = 'log')
```

In [20]:
```
model_LR.fit(data[names], Y)
```

Out[20]:
```
SGDClassifier(loss='log')
```

In [21]:
```
imp_LR_std = model_LR.coef_
print(imp_LR_std)
```

```
[[-3.59743861  4.07524765 19.02553695]]
```

In [22]:
```
model_SVM = linear_model.SGDClassifier()
```

In [23]:
```
model_SVM.fit(data[names], Y)
```

Out[23]:
```
SGDClassifier()
```

In [24]:
```
imp_SVM_std = model_SVM.coef_
```

```
print(imp_SVM_std)
```

```
[[-2.40630821 -0.12672137 16.7315999 ]]
```

Observation: 1. Applying SGDclassifier for logistic regression and SVM before standadization the feature impornce like f3 more important than f1 and f2 f2 is least important f1 is second most important feature 2. Applying SGDclassifier for logistic regression and SVM after standadization the feature impornce like values of the coeffients are readable like before f3 more important than f1 and f2 But now f2 become important than f1 f1 is now least important

In [ ]: