In [6]:
```
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 [122…
```
data = pd.read_csv('task_b.csv')
data=data.iloc[:,1:]
```

In [123…
```
data.head()
```

Out[123…

	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 [124…
```
data.corr()['y']
```

Out[124…
```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

In [125…
```
data.std()
```

Out[125…
```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

In [133…
```
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 [155…
```
# Applying logistic regression  without standarizing feature
from sklearn import linear_model
lin_svc_lg = linear_model.SGDClassifier(loss='log')
lin_svc_lg.fit(X,Y)

feature_names = ['f1','f2','f3']
feat_import = sorted(zip(lin_svc_lg.coef_[0], feature_names),reverse=True)

print("feature importance in decending order")
for coef, feat in feat_import:
    print ( coef, feat)
```

```
feature importance in decending order
10244.95099680913 f3
9225.318406193299 f1
-5582.346977703901 f2
```

Observation

logistic regression without standarizing feature :->

1. As we know weight coeficient is inversly proportion to variance, so weight to feature will be less if variance is more and also data is not standarized so the model can be impact by the data. But after running the model for 10 times , i have observed the feature importance is found to be inversely proportional to variance.

In [156…
```
# Applying SVM  without standarizing feature
from sklearn import linear_model
lin_svc_lg = linear_model.SGDClassifier(loss='hinge')
lin_svc_lg.fit(X,Y)

feature_names = ['f1','f2','f3']
feat_import = sorted(zip(lin_svc_lg.coef_[0], feature_names),reverse=True)
```

```
print("feature importance in decending order")
for coef, feat in feat_import:
    print ( coef, feat)
```

```
feature importance in decending order
10127.486241934781 f3
9866.074356996713 f1
4637.689960093515 f2
```

Observation

SVM without standarizing feature :->

1.same observation i have found in this case.

In [108…

```
# Applying logistic regression  with standarizing feature
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_stand = sc.fit_transform(X)

lin_svc_lg = linear_model.SGDClassifier(loss='log',,random_state = True)
lin_svc_lg.fit(X_stand,Y)

feature_names = ['f1','f2','f3']
feat_import = sorted(zip(lin_svc_lg.coef_[0], feature_names),reverse=True)

print("feature importance in decending order")
for coef, feat in feat_import:
    print ( coef, feat)
```

```
feature importance in decending order
11.942874476702952 f3
0.4142658807766934 f1
-0.06285984211624956 f2
```

Observation

logistic regression with standarizing feature :->

1. As the dataset is standarize , the model does not impact by outlier. So the feature corresponding to less variance has been given more weightage.

In [130…

```
# Applying SVM  with standarizing feature
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_stand = sc.fit_transform(X)

lin_svc_lg = linear_model.SGDClassifier(loss='hinge',random_state = True)
lin_svc_lg.fit(X_stand,Y)

feature_names = ['f1','f2','f3']
feat_import = sorted(zip(lin_svc_lg.coef_[0], feature_names),reverse=True)

print("feature importance in decending order")
i=0
for coef, feat in feat_import:
  if i<=10:
```

```
    i=i+1
    print ( coef, feat)
```

```
feature importance in decending order
14.907352035215503 f3
-2.0927784094761073 f1
-2.6966881696273415 f2
```

Observation

SVM with standarizing feature :->

1. As the dataset is standarize , the model does not impact by outlier. So the feature corresponding to less variance has been given more weightage.