In [11]:
```
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 [12]:
```
data = pd.read_csv('task_b.csv')
data=data.iloc[:,1:]
```

In [13]:
```
data.head()
```

Out[13]:

	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 [14]:
```
data.corr()['y']
```

Out[14]:
```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

In [15]:
```
data.std()
```

Out[15]:
```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

In [16]:
```
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

TASK 01. A

1. Apply Logistic regression(SGDClassifier with logloss) on 'data' and check the feature importance

In [27]:
```
from sklearn.linear_model import SGDClassifier
import itertools

clf_SDG_loss = SGDClassifier(loss='log')
clf_SDG_loss.fit(X, Y)

features_importance = dict(zip(list(data.columns)[:-1], list(itertools.chain.from_iterable(list(clf_SDG_loss.coef_.tolist())))))


keymax = max(features_importance, key=features_importance.get)


print("Feature_Importance: {}".format(features_importance))
print("Most important feature without data standardization is {} .".format(keymax))
```

```
Feature_Importance: {'f1': 11594.880562504617, 'f2': -5995.434695690012, 'f3': 10886.277098714618}
Most important feature without data standardization is f1 .
```

TASK 01. B

1. Apply SVM(SGDClassifier with hinge) on 'data' and check the feature importance.

In [28]:
```
from sklearn.linear_model import SGDClassifier
import itertools

clf_SDG_loss = SGDClassifier(loss='hinge')
clf_SDG_loss.fit(X, Y)

features_importance = dict(zip(list(data.columns)[:-1], list(itertools.chain.from_iterable(list(clf_SDG_loss.coef_.tolist())))))


keymax = max(features_importance, key=features_importance.get)


print("Feature_Importance: {}".format(features_importance))
print("Most important feature without data standardization is {} .".format(keymax))
```

```
Feature_Importance: {'f1': 18985.148506920534, 'f2': -7414.958896978234, 'f3': 10108.85496768173}
Most important feature without data standardization is f1 .
```

OBSERVATION:

- In the above example, if we observe the variance / std. deviation of the features f1:488.195035, f2:10403.417325, f3: 2.926662 i.e. var(F2)>>var(F1)>>Var(F3) and both the models has given more importance to F3 which has extremely low variance s compared to other two features.
- As we havent perform data standardization/mean-centering before applying model, each feature would have different scale which could impact the results as many linear models consider distance between the points while calculating hyperplanes and therefore model would have given more importance to the feature with low variance i.e. F3.
- In gradient discent algoriths as well, when we havent perform data scaling/ standardization, randomly initiated point could lie far away from the minima which may leads to slower convergence.

TASK 02. A

1. A. 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

In [29]:
```
from sklearn.preprocessing import StandardScaler
clf_stand = StandardScaler()

#clf_stand.fit(X, Y)
#std_data = clf_stand.transform(X)

std_data = data.drop(['y'], axis=1)
std_data[['f1', 'f2', 'f3']] = clf_stand.fit_transform(std_data[['f1', 'f2', 'f3']])

clf_SDG_loss = SGDClassifier(loss='log')
clf_SDG_loss.fit(std_data, Y)
clf_SDG_loss.coef_

features_importance = dict(zip(list(data.columns)[:-1], list(itertools.chain.from_iterable(list(clf_SDG_loss.coef_.tolist())))))

keymax = max(features_importance, key=features_importance.get)


print("Feature_Importance: {}".format(features_importance))
print("Most important feature without data standardization is {} .".format(keymax))
```

```
Feature_Importance: {'f1': 0.09405264994852117, 'f2': -1.279090680788435, 'f3': 9.249586630524444}
Most important feature without data standardization is f3 .
```

TASK 02. B

1. 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 [26]:
```
#from sklearn.preprocessing import StandardScaler
#clf_stand = StandardScaler()

#clf_stand.fit(X, Y)
#std_data = clf_stand.transform(X)

std_data = data.drop(['y'], axis=1)
std_data[['f1', 'f2', 'f3']] = clf_stand.fit_transform(std_data[['f1', 'f2', 'f3']])


clf_SDG_loss = SGDClassifier(loss='hinge')
clf_SDG_loss.fit(std_data, Y)
clf_SDG_loss.coef_

features_importance = dict(zip(list(data.columns)[:-1], list(itertools.chain.from_iterable(list(clf_SDG_loss.coef_.tolist())))))


keymax = max(features_importance, key=features_importance.get)


print("Feature_Importance: {}".format(features_importance))
print("Most important feature without data standardization is {} .".format(keymax))
```

```
Feature_Importance: {'f1': 0.5663865749858614, 'f2': 2.153225544753947, 'f3': 16.57343155479748}
Most important feature without data standardization is f3 .
```

OBSERVATION:

- After Standardization, all the features have same variance, feature f3 is still most important feature as it the weight assigned to this feature is higher than other two features and comparatively f1 seems to be less important as the weight associated to this feature is lesser than others.
