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.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import MinMaxScaler
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings("ignore")
```

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.drop('y', axis = 1, inplace = False).std()
```

Out[5]:

```
f1      488.195035
f2    10403.417325
f3        2.926662
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

Task1

In [7]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

x_train, x_test, y_train_, y_test = train_test_split(X, Y, stratify = Y, test_size = 0.3)

'''
penalty = {'l2', 'l1', 'elasticnet'}, default='l2'
'l1' and 'elasticnet' might bring sparsity to the model (feature selection) not achievable with 'l2'.
'''

sgd_log_clf = SGDClassifier(loss = 'log', penalty = 'l1', max_iter=1000, tol=0.01, n_jobs = -1)
sgd_log_clf.fit(X, Y)
log_loss_import = sgd_log_clf.coef_

sgd_hinge_clf = SGDClassifier(loss = 'hinge', penalty = 'l1', max_iter=1000, tol=0.01, n_jobs = -1)
sgd_hinge_clf.fit(X, Y)
hinge_loss_import = sgd_hinge_clf.coef_

# print(log_loss_import)
# print(hinge_loss_import)
```

Task2

In [8]:

```
std_data = data.drop('y', axis = 1)

for key in std_data.keys():
    col_mean = data[key].mean()
    col_std = data[key].std()
    std_data[key] = ((std_data[key] - col_mean) / col_std)
```

In [9]:

```
sgd_log_clf = SGDClassifier(loss = 'log', penalty = 'l1', max_iter=1000, tol=0.01, n_jobs = -1)
sgd_log_clf.fit(std_data, Y)
std_log_loss_import = sgd_log_clf.coef_

sgd_hinge_clf = SGDClassifier(loss = 'hinge', penalty = 'l1', max_iter=1000, tol=0.01, n_jobs = -1)
sgd_hinge_clf.fit(std_data, Y)
std_hinge_loss_import = sgd_hinge_clf.coef_

# print(std_log_loss_import)
# print(std_hinge_loss_import)
```

Make sure you write the observations for each task, why a particular feautre got more importance than others

In [10]:

```
data.describe()
```

Out[10]:

	f1	f2	f3	y
count	200.000000	200.000000	200.000000	200.000000
mean	10.180031	1299.986739	5.001840	0.500000
std	488.195035	10403.417325	2.926662	0.501255
min	-1662.579110	-29605.563847	0.076763	0.000000
25%	-303.220980	-5626.637315	2.508042	0.000000
50%	4.684317	2611.405803	5.029256	0.500000
75%	312.239850	8075.864754	7.436617	1.000000
max	1130.609573	24131.360720	9.933769	1.000000

In [11]:

```
std_data.describe()
```

Out[11]:

	f1	f2	f3

count	2.000000e+02	2.000000e+02	2.000000e+02
mean	-4.440892e-17	-8.881784e-18	-2.609024e-16
std	1.000000e+00	1.000000e+00	1.000000e+00
min	-3.426416e+00	-2.970711e+00	-1.682831e+00
25%	-6.419586e-01	-6.658028e-01	-8.520965e-01
50%	-1.125721e-02	1.260566e-01	9.367869e-03
75%	6.187278e-01	6.513127e-01	8.319299e-01
max	2.295045e+00	2.194603e+00	1.685172e+00

In [12]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
'''score(X, y[, sample_weight])          Return the mean accuracy on the given test data and labels'''

print('Mean accuracy score WITHOUT Standardization')
print('='*43)

print(f'With SGD & logloss\t\t: {sgd_log_clf.score(X, Y)}')
for index, coeff in enumerate(log_loss_import[0]):
    print(f'f{index + 1} coefficient {round(log_loss_import[0][index], 4)}')

print(f'\nWith SGD & hinge loss\t\t: {sgd_hinge_clf.score(X, Y)}')
for index, coeff in enumerate(hinge_loss_import[0]):
    print(f'f{index + 1} coefficient {round(hinge_loss_import[0][index], 4)}')
```

```
Mean accuracy score WITHOUT Standardization
===========================================
With SGD & logloss              : 0.505
f1 coefficient 1275.6885
f2 coefficient -8887.6494
f3 coefficient 25740.3247

With SGD & hinge loss           : 0.505
f1 coefficient 7645.5297
f2 coefficient -17392.0799
f3 coefficient 20279.4667
```

- Here in both the models the `f3` is getting highest importance than `f1` and `f2`.
- The coefficient values are larger because of we haven't standardized the values.
- Because of this higher coefficient values it is not interpretable.
- From both models the `f2` has high variance and thus those became the least important features.
- `f3` has low variance and thus this became the most important features.
- The high variance features can lead us to overfitting.
- SVM model is performing slightly better than logistic regression.

In [13]:

```
print('Mean accuracy score WITH Standardization')
print('='*40)
print(f'With SGD & logloss\t\t: {sgd_log_clf.score(std_data, Y)}')
for index, coeff in enumerate(std_log_loss_import[0]):
    print(f'f{index + 1} coefficient {round(std_log_loss_import[0][index], 4)}')

print(f'\nWith SGD & hinge loss\t\t: {sgd_hinge_clf.score(std_data, Y)}')
for index, coeff in enumerate(std_hinge_loss_import[0]):
    print(f'f{index + 1} coefficient {round(std_hinge_loss_import[0][index], 4)}')
```

```
Mean accuracy score WITH Standardization
========================================
With SGD & logloss              : 0.895
f1 coefficient -1.325
f2 coefficient 0.0
f3 coefficient 9.1889

With SGD & hinge loss           : 0.905
f1 coefficient -4.2437
f2 coefficient 0.0
f3 coefficient 12.9818
```

- Here in both the models the `f3` is getting highest importance than `f1` and `f2`.
- By doing the statndardization we are not preserving the variance of the data.
- The standardisation helps to make the mean to zero and variance to 1.
- By doing standardizarion the variance became 1 for all features.

- This helped the algorithm to predict the class in much more efficient way.
- Due to that the mean accuracy score after standardization improved by a good margin.
- From both models the `f2` has high variance and thus this became the least important features.
- `f3` has low variance and thus this became the most important features.
- SVM model is performing slightly better than logistic regression.

https://youtu.be/0HOqOcln3Z4