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.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)
```

```
(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
```

TASK 1

In [7]:

```
scaler = MinMaxScaler()
X_m = scaler.fit_transform(X,Y)
```

In [8]:

```
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(loss='log')
model.fit(X_m,Y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=
5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and d
efault tol will be 1e-3.
```

Out[8]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

In [9]:

```
coef = model.coef_
for f,imp in enumerate(coef[0]):
    print(f"feature {f}: importance {imp}")
```

```
feature 0: importance -7.985713233605663
feature 1: importance -2.5698872210213053
feature 2: importance 43.151201617877824
```

In [10]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=
5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and d
efault tol will be 1e-3.
```

Out[10]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

In [11]:

```
coef = clf.coef_
for f,imp in enumerate(coef[0]):
    print(f"feature {f}: importance {imp}")
```

```
feature 0: importance -7.412482042849076
feature 1: importance 3.959331423878054
feature 2: importance 53.86986397425413
```

TASK 2

In [12]:

```
scaler = StandardScaler()
X_sc = scaler.fit_transform(X,Y)
```

using sgd classifier with log loss that is equivalent to logistic regression

In [13]:

```
model = SGDClassifier(loss='log')
model.fit(X_sc,Y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=
5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and d
efault tol will be 1e-3.
```

Out[13]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

Feature Importance

In [14]:

```
coef = model.coef_
for f,imp in enumerate(coef[0]):
    print(f"feature {f}: importance {imp}")
```

```
feature 0: importance 1.9379737585908223
feature 1: importance -6.526061145185106
feature 2: importance 30.024332147066083
```

In [15]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=
5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and d
efault tol will be 1e-3.
```

Out[15]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
              n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=None,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

In [16]:

```
coef = clf.coef_
for f,imp in enumerate(coef[0]):
    print(f"feature {f}: importance {imp}")
```

```
feature 0: importance -6.941473351924469
feature 1: importance 7.890197742428744
feature 2: importance 37.2286321904762
```

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

Feature 3 has high value for correlation and feauture 2 lowest. This shows values in feature 3 column are highly correlated

TASK 1 observations

In logistic regression, feature importance in descending order : feature 2,1,0. This is because the weight values of feature 2 is 43.15, feature 1 -2.56 and feature 0 is -7.98. The weight values of feature 2 is highest, the contribution made by it is higher than 1 and 0. Since the coefficient value corresponding to feature 2 is high and positive, the probability of query point belonging to positive class is also high

Even in svm, the order of feature importance in descending order is 2,1,0. Here also the weight or ceofficient values are used to determine feature importances.

Task 2 - After standardization of data

Data should be stadardized, because in both models distance is used to calculate the distance of datapoint from hyperplane.

After standardization, coefficients values change for both models. Feature importance for svm stays same, but the order for logistic regression changes (in descending order of importance) - feature 2,0,1

In [ ]: