```
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, SGDClassifier
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)
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
%cd /content/drive/My Drive/Appliedai colab/Assignment 8 - Linear models
```

```
Mounted at /content/drive
/content/drive/My Drive/Appliedai colab/Assignment 8 - Linear models
```

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

```
data.head()
```

	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

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

```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

```
data.std()
```

```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

```
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

The features aren't scaled so the features with larger values will get more feature importance.

```
cols =  ['f1','f2','f3']
```

```
np.array([cols]).T
```

```
array([['f1'],
       ['f2'],
       ['f3']], dtype='<U2')
```

Task 1

Logistic regression(SGDClassifier with logloss)

```
clf = SGDClassifier(loss= 'log',n_jobs = -1)
clf.fit(X, Y)
clf.coef_
```

```
array([[  -84.27630295,  2485.28006292, 11331.71843936]])
```

```
abs(clf.coef_.T)
```

```
array([[   84.27630295],
       [ 2485.28006292],
       [11331.71843936]])
```

```
data = {'feature': np.array([cols])[0], 'weights': abs(clf.coef_[0])}
feature_importance = pd.DataFrame(data)
```

```
feature_importance.sort_values(by='weights', ascending=False)
```

	feature	weights
2	f3	11331.718439
1	f2	2485.280063
0	f1	84.276303

- As we have high variance in the features we can see that the feature weights are also very high

SVM(SGDClassifier with hinge)

```
clf = SGDClassifier(loss= 'hinge',n_jobs = -1)
clf.fit(X, Y)
abs(clf.coef_)
```

```
array([[11650.65334127,  6338.07610453,  9759.8560015 ]])
```

```
data = {'feature': np.array([cols])[0], 'weights': abs(clf.coef_[0])}
feature_importance = pd.DataFrame(data)
feature_importance.sort_values(by='weights', ascending=False)
```

	feature	weights
0	f1	11650.653341
2	f3	9759.856001
1	f2	6338.076105

- Same as before, due to lack of scaling/standardisation and having high variance data our weights are large
- Both of the above models are predicting different feature importance due to this

Task 2

```
X = StandardScaler().fit_transform(X)
```

```
pd.DataFrame(X).head()
```

	0	1	2
0	-0.423126	-1.555602	0.181651
1	-2.520394	-0.517290	-0.200648
2	-0.002139	0.300020	-1.567659
3	0.726209	1.365930	-1.338565
4	-1.599662	-0.892703	-1.072608

Logistic regression(SGDClassifier with logloss) on 'data' after standardization

```
clf = SGDClassifier(loss= 'log',n_jobs = -1)
clf.fit(X, Y)
abs(clf.coef_)
```

```
abs(clf.coef_)
```

```
array([[ 3.19919538,  0.29174776, 12.73351643]])
```

```
data = {'feature': np.array([cols])[0], 'weights': abs(clf.coef_[0])}
feature_importance = pd.DataFrame(data)
feature_importance.sort_values(by='weights', ascending=False)
```

	feature	weights
2	f3	12.733516
0	f1	3.199195
1	f2	0.291748

- After Column standardization the model is providing much smaller weights which are close
- Hence, we've obtained new importance sequence f3, f1, f2

```
clf = SGDClassifier(loss= 'hinge',n_jobs = -1)
clf.fit(X, Y)
abs(clf.coef_)
```

```
array([[ 2.11929181,  1.74917378, 19.6340757 ]])
```

```
data = {'feature': np.array([cols])[0], 'weights': abs(clf.coef_[0])}
feature_importance = pd.DataFrame(data)
feature_importance.sort_values(by='weights', ascending=False)
```

	feature	weights
2	f3	19.634076
0	f1	2.119292
1	f2	1.749174

- The feature importance sequence is still f3, f1, f2

```
# https://stats.stackexchange.com/questions/146277/hinge-loss-vs-logistic-loss-advantages-and-disadvantages-limitations
```