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 fe
* 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 fea
    2. Apply SVM(SGDClassifier with hinge) on 'data' and check the feature importance

> Task2:
    1. Apply Logistic regression(SGDClassifier with logloss) on 'data' after standardiza
       i.e standardization(data, column wise): (column-mean(column))/std(column) and che
    2. Apply SVM(SGDClassifier with hinge) on 'data' after standardization
       i.e standardization(data, column wise): (column-mean(column))/std(column) and che
```

```
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)
```

```
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

Task1

```
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,)
```

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

```
from sklearn import linear_model
clf = linear_model.SGDClassifier(loss='log',random_state=15)
clf.fit(X, Y)
#feature importance is weight itself.
print(clf.coef_)
```

```
[[  3925.14601273 -16033.05764291  10502.94022174]]
```

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

```
clf = linear_model.SGDClassifier(loss='hinge',random_state=15)
clf.fit(X, Y)
#feature importance is weight itself
print(clf.coef_)
```

```
[[-1441.65036452 -3083.88512888 10638.5348658 ]]
```

Observation

1. in Logistic regression feature importance f2 >f3 >f1 because f2 absolute weight f2> f3 >f1
2. in SVM feature importance f3 >f2 >f1 because f2 absolute weight f3> f2 >f1

Task2:

features standrization

```
#https://stackoverflow.com/questions/49641707/standardize-some-columns-in-python-pandas-da

data[['f1', 'f2','f3']] = StandardScaler().fit_transform(data[['f1', 'f2','f3']])
data.head()
```

	f1	f2	f3	y
0	-0.423126	-1.555602	0.181651	1.0
1	-2.520394	-0.517290	-0.200648	1.0
2	-0.002139	0.300020	-1.567659	0.0
3	0.726209	1.365930	-1.338565	0.0
4	-1.599662	-0.892703	-1.072608	0.0

```
X=data[['f1','f2','f3']].values
Y=data['y'].values
print(X.shape)
print(Y.shape)
```

```
(200, 3)
(200,)
```

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

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

```
data.std()
```

```
f1    1.002509
f2    1.002509
f3    1.002509
```

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

```
from sklearn import linear_model
clf = linear_model.SGDClassifier(loss='log',random_state=15)
clf.fit(X, Y)
#feature importance is weight itself
print(clf.coef_)
```

```
[[-0.29741788 -0.66973479 10.35436789]]
```

2. Apply SVM(SGDClassifier with hinge) on 'data' after standardization

```
clf = linear_model.SGDClassifier(loss='hinge',random_state=15)
clf.fit(X, Y)
#feature importance is weight itself
print(clf.coef_)
```

```
[[ 2.23347737  0.46842383 22.39791493]]
```

observation

1. SVM and Logistic regression methods are based on distance so it is required to scale variables prior to running final model
2. in Logistic regression feature importance f3 >f2 >f1 because f2 absolute weight f3 > f2 >f1
3. in SVM feature importance f3 >f2 >f1 because f2 absolute weight f3> f2 >f1