In [134]:

```
from google.colab import drive
drive.mount('content/')
```

```
Drive already mounted at content/; to attempt to forcibly remount, call drive.mount("content/", fo
rce_remount=True).
```

In [135]:

```
path='/content/content/My Drive/AAIC/8_LinearModels-20200808T170255Z-001/8_LinearModels/'
```

In [136]:

```
#!pip install plotly --upgrade
```

In [137]:

```
#https://stackoverflow.com/questions/47230817/plotly-notebook-mode-with-google-
colaboratory/47230966
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></scr
ipt>'''))
  init_notebook_mode(connected=False)
```

In [138]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import plotly
import plotly.figure_factory as ff
import plotly.graph_objs as go
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from plotly.offline import init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [139]:

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

In [140]:

```
data.head()
```

Out[140]:

	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 [141]:

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

Out[141]:

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

In [142]:

```
data.std()
```

Out[142]:

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

In [143]:

```
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

Task1

In [144]:

```
def draw_hyperplane(coef,d,x_min,x_max,y_min,y_max):

  print("Hyperplane coefficients and intercept are ",coef,d)
  z=[0]*4
  z[0] = list(-1*(d+((coef[0][0]*x_min)+(coef[0][1]*y_min)))/(coef[0][2]))
  z[1] = list(-1*(d+((coef[0][0]*x_min)+(coef[0][1]*y_max)))/(coef[0][2]))
  z[2] = list(-1*(d+((coef[0][0]*x_max)+(coef[0][1]*y_min)))/(coef[0][2]))
  z[3] = list(-1*(d+((coef[0][0]*x_max)+(coef[0][1]*y_max)))/(coef[0][2]))

  z1=[]
  x=[x_min,x_min,x_max,x_max]
  y=[y_min,y_max,y_min,y_max]
  for k in range(len(z)):
    z1.append(z[k][0])

  return x,y,z1
```

Logistic Regression

In [145]:

```
clf = SGDClassifier(loss='log')
clf.fit(X,Y)
w_lr = clf.coef_
b_lr = clf.intercept_
print("Hyperplane coefficients and intercept are",w_lr,b_lr)
```

```
Hyperplane coefficients and intercept are [[6032.53771713 2154.95005917 9908.7994988 ]] [-
4.47280977]
```

In [146]:

```
enable_plotly_in_cell()

layout = go.Layout()
fig = go.Figure(layout = layout)

fig.add_trace(go.Scatter3d(
   x = X[:,0], y = X[:,1], z = X[:,2],mode = 'markers', marker = dict(
      size = 12,
      color = Y,
      colorscale = 'Viridis'
      )))
x1,y1,z1 = draw_hyperplane(w_lr,b_lr,X[:,0].min(axis=0),X[:,0].max(axis=0),X[:,1].min(axis=0),X[:,1
].max(axis=0))
fig.add_trace(go.Mesh3d(x=x1, y=y1, z=z1, color='green', opacity=0.6))
fig.update_layout(
    title = '3D Scatter plot of features and hyperplane with LR',
    scene = dict(
        xaxis = dict(nticks=10, title = 'feature-f1',range=[-2000,2000]),
        yaxis = dict(nticks=10, title = 'feature-f2'),
        zaxis = dict(nticks=10, title = 'feature-f3',range=[-2,15])
        ),
    margin=dict(l=5,r=5,b=5))
fig.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

SVM

In [147]:

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

w_svm = clf.coef_
b_svm = clf.intercept_

print(w_svm,b_svm)
```

```
[[  6535.48144601 -28342.50214468  10869.6377527 ]] [37.50079002]
```

In [148]:

```
enable_plotly_in_cell()

layout = go.Layout()
fig = go.Figure(layout = layout)

fig.add_trace(go.Scatter3d(
   x = X[:,0], y = X[:,1], z = X[:,2],mode = 'markers', marker = dict(
      size = 12,
      color = Y,
      colorscale = 'Viridis'
      )))
x1,y1,z1 = draw_hyperplane(w_svm,b_svm,X[:,0].min(axis=0),X[:,0].max(axis=0),X[:,1].min(axis=0),X[:
,1].max(axis=0))

fig.add_trace(go.Mesh3d(x=x1, y=y1, z=z1, color='green', opacity=0.6))
fig.update_layout(
    title = '3D Scatter plot of features and hyperplane with SVM',
    scene = dict(
        xaxis = dict(nticks=10, title = 'feature-f1',range=[-2000,2000]),
        yaxis = dict(nticks=10, title = 'feature-f2'),
        zaxis = dict(nticks=10, title = 'feature-f3',range=[-2,15])
        ),
    margin=dict(l=5,r=5,b=5))
fig.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

In [149]:

```
print("Logistic Regression Hyperplane coeficients are",w_lr)
print("SVM Hyperplane coeficients are",w_svm)
```

```
Logistic Regression Hyperplane coeficients are [[6032.53771713 2154.95005917 9908.7994988 ]]
SVM Hyperplane coeficients are [[  6535.48144601 -28342.50214468  10869.6377527 ]]
```

From the above coefficients we can observe that as we assume features are independent

1. Feature f1 and f3 in LR and f2 and f3 in SVM are more important and this varies due to random shuffling of data due to SGD.
2. This is due to W1 and W3 in LR and W2 and W3 in SVM are the greatest values among the coefficients.

var(F2)>>var(F1)>>Var(F3) as observed from the data. Due to this hyperplane is highly tilted towards the f2 and hyperplane is lying in the plane of f2 and f1. Both svm and LR are highly impacted due to variatoin in the feature variance. Thus the formed hyperplane is misclassifying most of the points and thereby giving accuracy < 50%.

Task2

In [150]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X = scaler.fit_transform(X)
```

Logistic Regression

In [151]:

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

w_lr = clf.coef_
b_lr = clf.intercept_

print("Hyperplane coefficients and intercept are",w_lr,b_lr)
```

```
Hyperplane coefficients and intercept are [[-0.22893974 -1.42629778 10.01841113]] [-0.04279428]
```

In [152]:

```
enable_plotly_in_cell()

layout = go.Layout()
fig = go.Figure(layout = layout)

fig.add_trace(go.Scatter3d(
   x = X[:,0], y = X[:,1], z = X[:,2],mode = 'markers', marker = dict(
      size = 12,
      color = Y,
      colorscale = 'Viridis'
      )))
x1,y1,z1 = draw_hyperplane(w_lr,b_lr,X[:,0].min(axis=0),X[:,0].max(axis=0),X[:,1].min(axis=0),X[:,1
].max(axis=0))

fig.add_trace(go.Mesh3d(x=x1, y=y1, z=z1, color='green', opacity=0.6))
fig.update_layout(
    title = '3D Scatter plot of Standardized features and hyperplane with LR',
    scene = dict(
        xaxis = dict(nticks=10, title = 'feature-f1'),
        yaxis = dict(nticks=10, title = 'feature-f2'),
        zaxis = dict(nticks=10, title = 'feature-f3')
        ),
    margin=dict(l=5,r=5,b=5))
fig.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

SVM

In [153]:

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

w_svm = clf.coef_
b_svm = clf.intercept_

print("Hyperplane coefficients and intercept are",w_svm,b_svm)
```

```
Hyperplane coefficients and intercept are [[-5.37250783 -3.1483992  14.93085923]] [-2.61383455]
```

In [154]:

```
enable_plotly_in_cell()

layout = go.Layout()
fig = go.Figure(layout = layout)

fig.add_trace(go.Scatter3d(
   x = X[:,0], y = X[:,1], z = X[:,2],mode = 'markers', marker = dict(
      size = 12,
      color = Y,
      colorscale = 'Viridis'
      )))
x1,y1,z1 = draw_hyperplane(w_svm,b_svm,X[:,0].min(axis=0),X[:,0].max(axis=0),X[:,1].min(axis=0),X[:
,1].max(axis=0))

fig.add_trace(go.Mesh3d(x=x1, y=y1, z=z1, color='green', opacity=0.6))
fig.update_layout(
    title = '3D Scatter plot of Standardized features and hyperplane with SVM',
    scene = dict(
        xaxis = dict(nticks=10, title = 'feature-f1'),
        yaxis = dict(nticks=10, title = 'feature-f2'),
        zaxis = dict(nticks=10, title = 'feature-f3')
        ),
    margin=dict(l=5,r=5,b=5))
fig.show()
fig.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

In [155]:

```
print("Logistic Regression Hyperplane coeficients are",w_lr)
print("SVM Hyperplane coeficients are",w_svm)
```

```
Logistic Regression Hyperplane coeficients are [[-0.22893974 -1.42629778 10.01841113]]
SVM Hyperplane coeficients are [[-5.37250783 -3.1483992  14.93085923]]
```

From the above coefficients we can observe that as we assume features are independent

1. Feature f3 in LR and SVM is more important. This is due to W3 in LR and SVM is the greatest value among the coefficients. As feature standardization is done, feature importance with weights is correctly interpreted.
2. Afte standardization, we can observe that hyperplane is correctly classifying the points and thereby giving accuracy more than 90%.

Conclusion

From the above assignment, we should understand that feature standardization is compulsory for obtaining the correct hyperplane coefficients and therby obtaining correct classification and best accuracy.