8B_LR_SVM

January 26, 2020

```
In [1]: import numpy as np
        import pandas as pd
        import plotly
        import plotly.figure_factory as ff
        import plotly.graph_objs as go
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings("ignore")
        from sklearn.linear_model import LogisticRegression
        from sklearn.linear_model import 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)
```

```
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
```

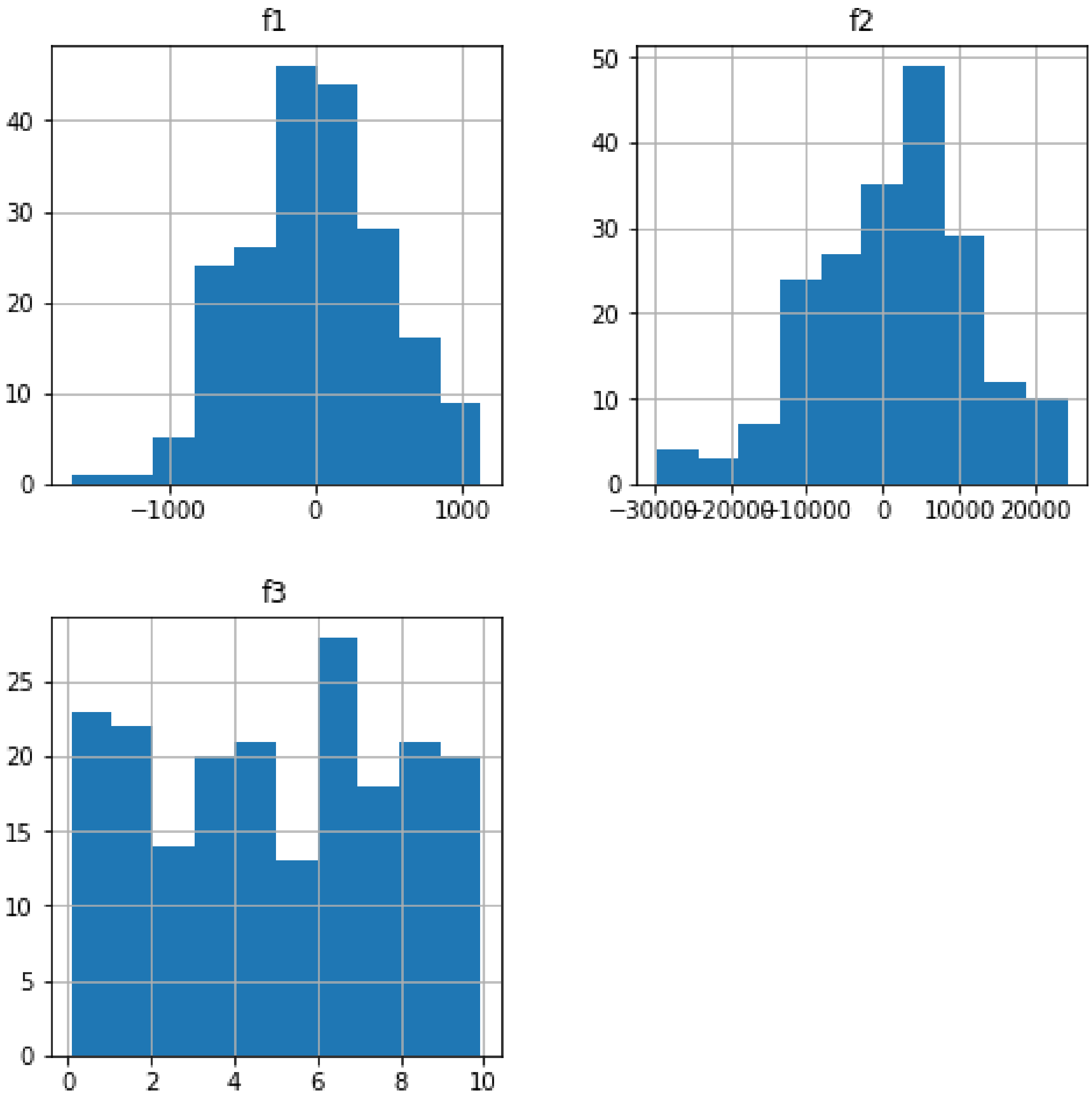
```
In [5]: data.std()
```

```
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,)
```

```
In [7]: fig = plt.figure(figsize = (8,8))
        ax = fig.gca()
        data[['f1','f2','f3']].hist(ax=ax)
        plt.show()
```



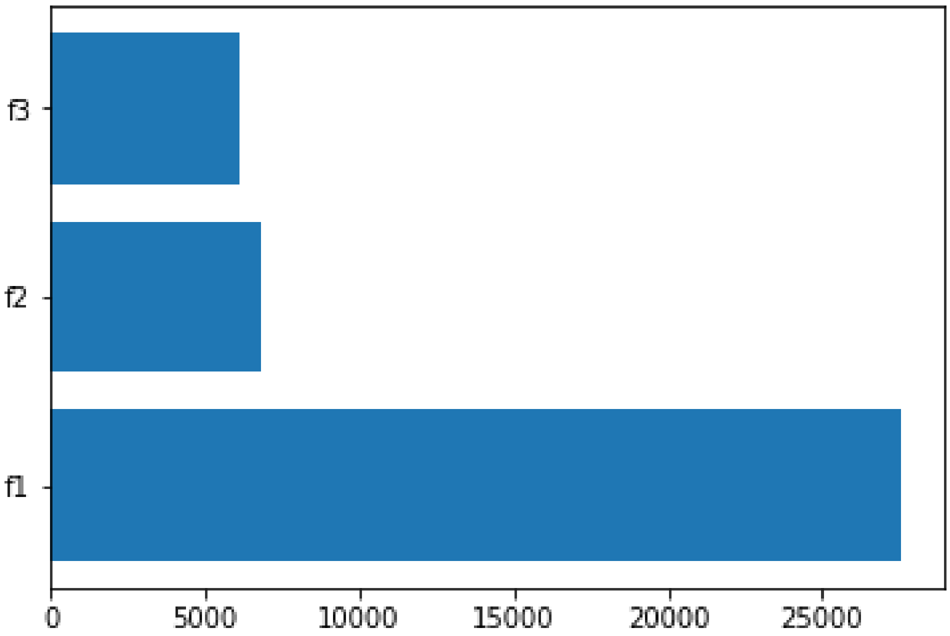
Make sure you write the observations for each task, why a particular feautre got more importance than others

```
In [8]: features_names = ['f1', 'f2', 'f3']
        def f_importances(coef, names):
            imp = coef
            imp,names = zip(*(zip(imp,names)))
            plt.barh(range(len(names)), imp, align='center')
            plt.yticks(range(len(names)), names)
            plt.show()
```

```
In [9]: cfg = SGDClassifier(loss="log")
        cfg.fit(X,Y)
        cfg.coef_,cfg.intercept_
```

```
Out[9]: (array([[27567.02012548,  6819.27989405,  6130.35161622]]),
         array([-48.49987675]))
```

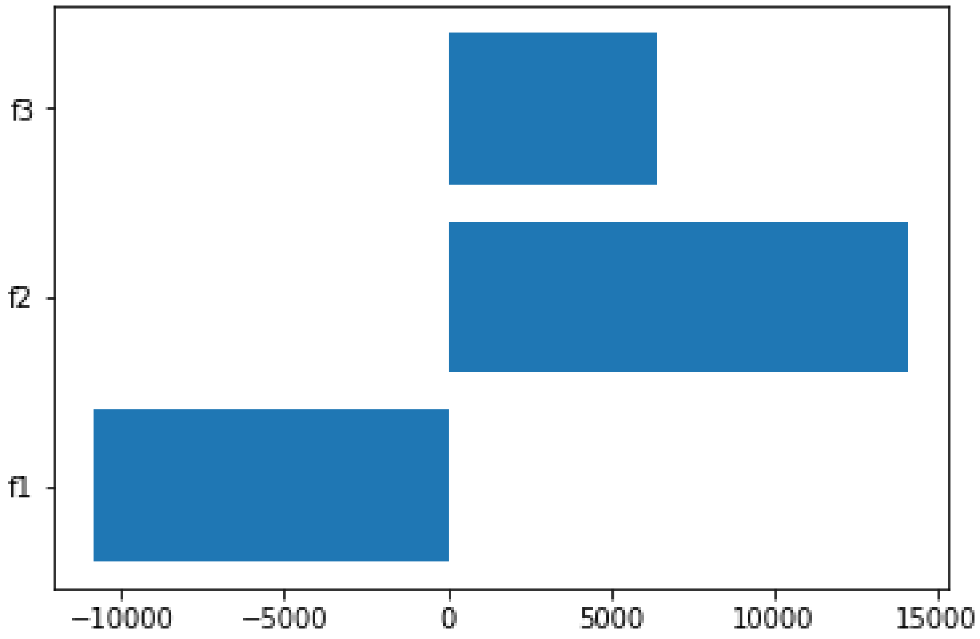
```
In [10]: f_importances(cfg.coef_[0],features_names)
```



```
In [11]: cfg = SGDClassifier(loss="hinge")
         cfg.fit(X,Y)
         cfg.coef_,cfg.intercept_
```

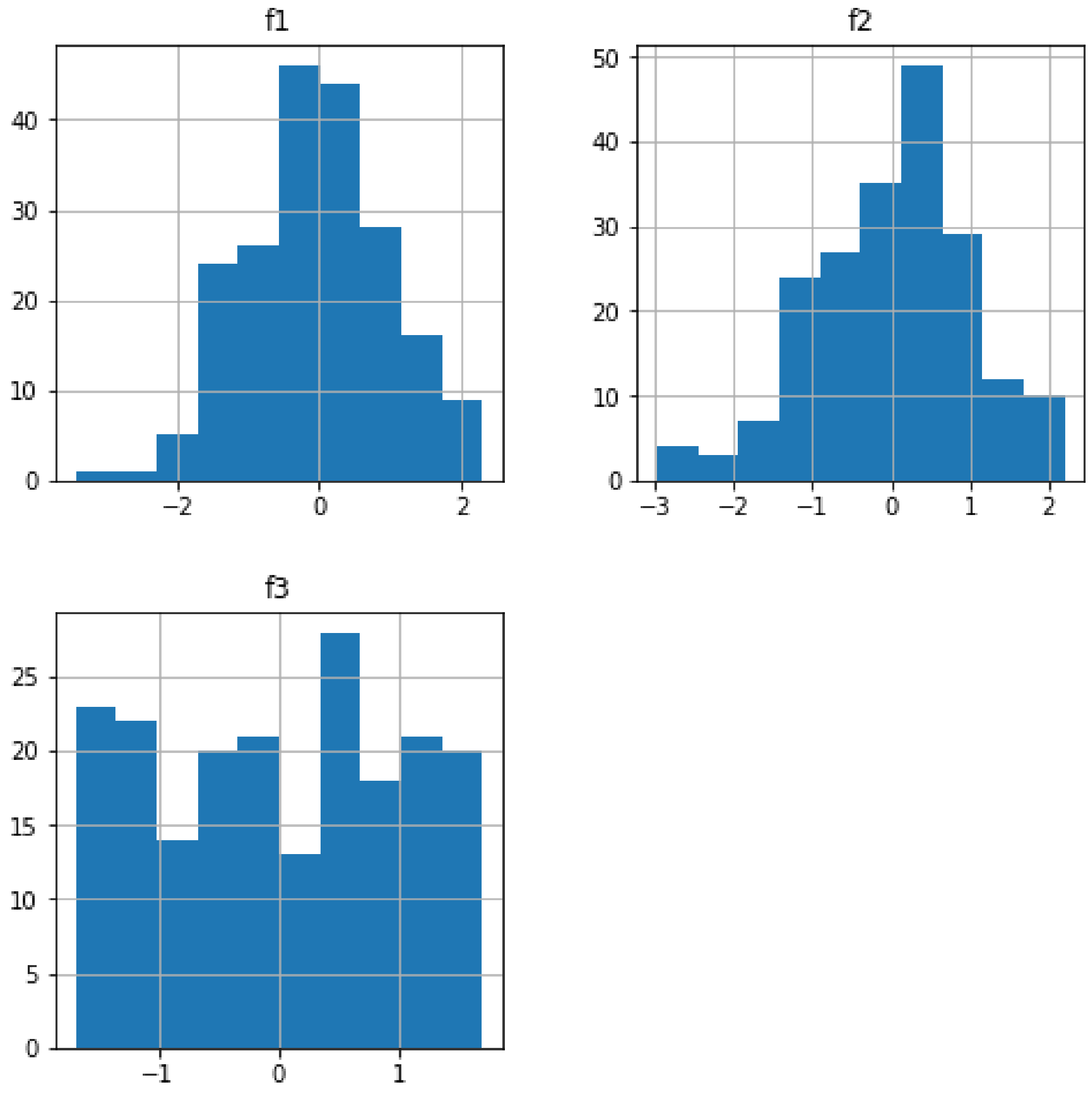
```
Out[11]: (array([[-10822.9527317 ,  14116.68388851,   6464.78235998]]),
          array([84.62459666]))
```

```
In [12]: f_importances(cfg.coef_[0],features_names)
```



```
In [13]: scaler = StandardScaler().fit(X)
         X_stand = scaler.transform(X)

In [14]: d = {'f1': X_stand[:,0], 'f2':X_stand[:,1],'f3': X_stand[:,2],'y': Y}
         data_stand = pd.DataFrame(data=d)
         fig = plt.figure(figsize = (8,8))
         ax = fig.gca()
         data_stand[['f1','f2','f3']].hist(ax=ax)
         plt.show()
```



```
In [15]: data_stand.corr()['y']
```

```
Out[15]: f1    0.067172
         f2   -0.017944
         f3    0.839060
         y     1.000000
         Name: y, dtype: float64
```

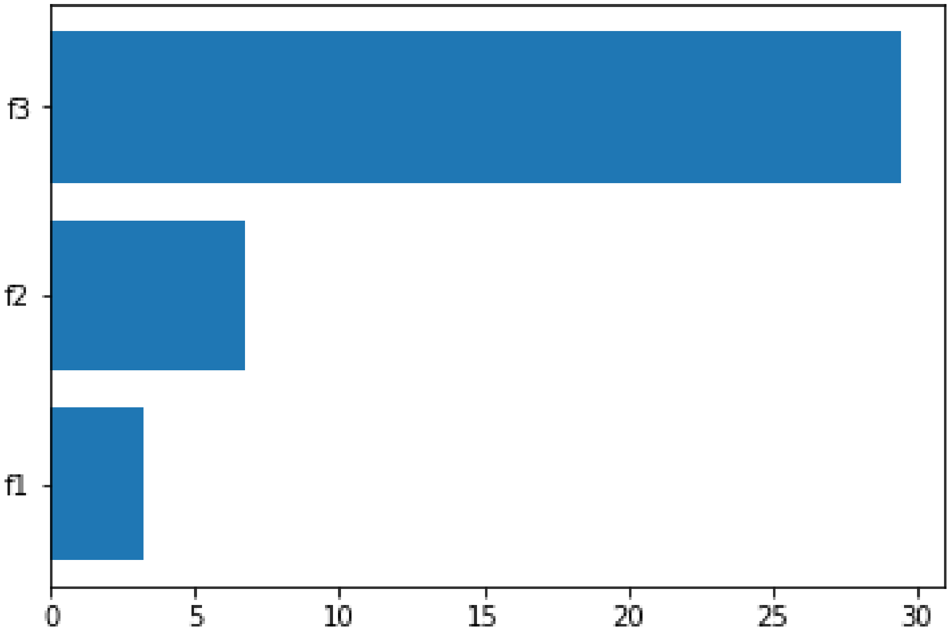
```
In [16]: data_stand.std()
```

```
Out[16]: f1    1.002509
         f2    1.002509
         f3    1.002509
         y     0.501255
         dtype: float64
```

```
In [17]: cfg = SGDClassifier(loss="log")
         cfg.fit(X_stand,Y)
         cfg.coef_,cfg.intercept_
```

```
Out[17]: (array([[ 3.22873377,  6.73267812, 29.45455474]]), array([9.75733498]))
```

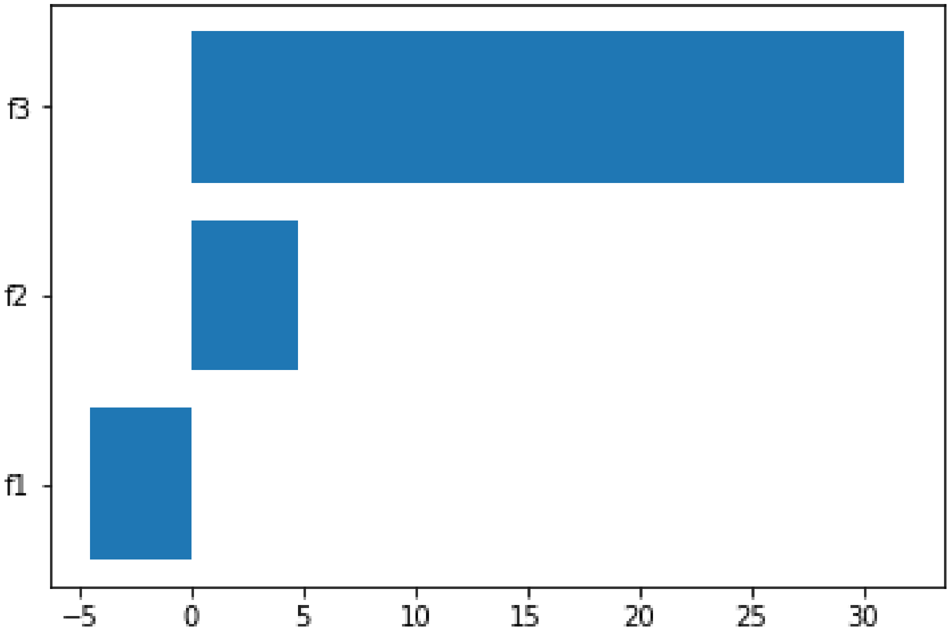
```
In [18]: f_importances(cfg.coef_[0],features_names)
```



```
In [19]: cfg = SGDClassifier(loss="hinge")
         cfg.fit(X_stand,Y)
         cfg.coef_,cfg.intercept_
```

```
Out[19]: (array([[-4.4536571 ,  4.78603897, 31.80348289]]), array([-5.66397733]))
```

```
In [20]: f_importances(cfg.coef_[0],features_names)
```


f3
f2
f1
−5
0
5
10
15
20
25
30