1 1.83 1.71 1.77 1.68 1.78 1.80 1.70 1.75 1.78 2 68177.69 66138.42 72981.88 74304.33 67549.66 69367.34 69169.41 73268.61 74465.84 72503.37 734	V92 V93 V94 V95 V96 V97 V98 V99 V100 Class 36.62 36.92 38.80 38.52 38.07 36.73 39.46 37.50 39.10 0 1.80 1.79 1.77 1.74 1.74 1.80 1.78 1.75 1.69 1
count 1212.000000 1212.00000 1222.00000 12222.00000 12222.00000 12222.00000 12222.00000 12222.000	48 8183.641543 8154.670066 8120.767574 8120.056815 8125.917409 8158.793812 8140.885421 95 18048.582159 17982.390713 17900.798206 17773.190621 17758.182403 17919.510371 17817.945646 00 0.650000 0.650000 0.620000 0.870000 0.900000 0.870000 0.880000 00 19.062500 19.532500 19.285000 19.197500 18.895000 19.237500 19.385000 00 290.850000 294.565000 295.160000 297.845000 295.420000 299.155000 293.355000
'V28', 'V29', 'V30', 'V31', 'V32', 'V33', 'V34', 'V35', 'V36', 'V37', 'V38', 'V39', 'V40' 3', 'V54', 'V55', 'V56', 'V57', 'V58', 'V59', 'V60', 'V61', 'V62', 'V63', 'V64', 'V65', ' 'V79', 'V80', 'V81', 'V82', 'V83', 'V84', 'V85', 'V86', 'V87', 'V88', 'V89', 'V90', 'V91' In [9]: df.shape Out[9]: (1212, 101) In [10]: df['Class'].value_counts() Out[10]: 0 606 1 606 Name: Class, dtype: int64 In [11]: df.groupby("Class").mean() Out[11]: V1 V2 V3 V4 V5 V6 V7 V6 Class 0 7913.333251 7825.339967 7902.497294 7857.032079 7775.610198 7875.436337 7804.166584 7722.324803	, 'V92', 'V93', 'V94', 'V95', 'V96', 'V97', 'V98', 'V99', 'V100', 'Class']
4 0 1207 1 1208 0 1209 1 1211 0 Name: Class, Length: 1212, dtype: int64 In [15]: X = df.drop(['Class'], axis=1) In [16]: X.shape Out[16]: (1212, 100) In [18]: X Out[18]: V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 0 39.02 36.49 38.20 38.85 39.38 39.74 37.02 39.53 38.81 38.79 1 1.83 1.71 1.77 1.77 1.68 1.78 1.80 1.70 1.75 1.78 2 68177.69 66138.42 72981.88 74304.33 67549.66 69367.34 69169.41 73268.61 74465.84 72503.37 3 44889.06 39191.86 40728.46 38576.36 45876.06 47034.00 46611.43 37668.32 40980.89 38466.15 4 5.70 5.40 5.28 5.38 5.27 5.61 6.00 5.38 5.34 5.87	
1208	33.18 32.76 35.03 32.89 31.91 33.85 35.28 32.49 32.83 34.82
In [21]: plt.plot(X.iloc[1,:]) plt.title('Hill'); Hill 20 19 18 17 from sklearn.preprocessing import StandardScaler In [23]: ss = StandardScaler() In [24]: X = ss.fit_transform(X) In [25]: X	
Out[25]: array([[-0.45248681, -0.45361784, -0.45100881,, -0.45609618, -0.45164274, -0.45545496], [-0.45455665, -0.45556372, -0.45302369,, -0.45821768, -0.45362255, -0.45755405], [3.33983604, 3.24466709, 3.58338069,, 3.5427869, 3.27907378, 3.74616847],, [0.11084204, 0.0505953, 0.04437307,, 0.12533312, 0.04456025, 0.06450317], [-0.45272112, -0.45369729, -0.45118691,, -0.45648861, -0.45190136, -0.4556951], [0.01782872, -0.02636986, 0.05196137,, 0.03036056, 0.01087365, 0.03123129]]) In [26]: X.shape Out[26]: (1212, 100) In [27]: from sklearn.model_selection import train_test_split In [28]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, stratify=y, random_In [29]: X_train.shape, X_test.shape, y_train.shape, y_test.shape Out[29]: ((848, 100), (364, 100), (848,), (364,)) In [30]: from sklearn.linear_model import LogisticRegression	state=72529)
<pre>In [31]: lr = LogisticRegression() In [32]: lr.fit(X_train, y_train) Out[32]: v LogisticRegression LogisticRegression() In [33]: y_pred=lr.predict(X_test) In [34]: y_pred_shape Out[34]: In [35]: y_pred Out[35]: array([0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,</pre>	
1. 0. 0. 0. 6. 5. 2. 3. 1. 0. 0. 0. 5. 5. 5. 1. 0. 0. 2. 0. 5. 6. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.	
1.	
Content Cont	V94 V95 V96 V97 V98 V99 V100
Out[49]: array([[0.49823599, 0.50176401]]) In []: In []:	