CMPE428 Assignment 3

Building Logistic Regression Classifiers

Imports

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import step

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
```

Import CSV

```
df = pd.read_csv('stdData.csv')
```

```
df.head()
```

	Label	V1	V2	V3	V4	V5	V6	V7	
0	positive	1.221400	128.101200	80.035036	35.431417	180.956968	42.944951	1.320305	-
1	negative	2.609743	85.891549	58.543681	14.454311	52.545356	33.426070	-0.786571	-
2	negative	2.682163	99.782456	68.000884	26.339627	71.578043	37.542894	0.534953	-
3	negative	3.196969	115.189168	65.307845	-0.539337	0.269863	20.857287	0.562433	-
4	positive	4.790932	144.487763	80.800220	18.937774	-0.033570	31.346055	0.789162	-

Task 1

Split Dataset with Equal Positives and Negatives

We can see the amount of negatives and positives with this command

```
df.Label.value_counts()
```

```
negative    195
positive    105
Name: Label, dtype: int64
```

Replacing Categorical with Binary

```
df["Label"] = df["Label"].replace({"positive":1,"negative":0})
```

```
df.head()
```

	Label	V1	V2	V3	V4	V5	V6	V7	
0	1	1.221400	128.101200	80.035036	35.431417	180.956968	42.944951	1.320305	-0.7
1	0	2.609743	85.891549	58.543681	14.454311	52.545356	33.426070	-0.786571	-1.6
2	0	2.682163	99.782456	68.000884	26.339627	71.578043	37.542894	0.534953	-3.3
3	0	3.196969	115.189168	65.307845	-0.539337	0.269863	20.857287	0.562433	-0.5
4	1	4.790932	144.487763	80.800220	18.937774	-0.033570	31.346055	0.789162	-4.0

Splitting Dataset Into 2, with Equal Amounts of Negative and Positive

We use stratify on y and split our dataframe into 2, so that we get equal distribution of negatives and positives.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.5, stratify = y)
```

```
y_train.value_counts()
```

```
0    97
1    53
Name: Label, dtype: int64
```

```
y_test.value_counts()
```

```
0    98
1    52
Name: Label, dtype: int64
```

We can see that numbers of 0 and 1 is equal across test and train.

Logistic Regression

We will use sklearn's logistic in order to get our accuracy, f1, recall and precision scores.

```
regression = LogisticRegression(solver = "liblinear")

model = regression.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Computing Scores

Scores are calculated and inserted into the dataframe to be easily compared with the other model's scores later on.

```
scores = {}
scores['Accuracy'] = accuracy_score(y_test, y_pred)
scores['F1'] = f1_score(y_test, y_pred)
scores['Recall'] = recall_score(y_test, y_pred)
scores['Precision'] = precision_score(y_test, y_pred)

scores = pd.DataFrame.from_dict(scores, orient='index', columns=['First DF'])
scores = scores.transpose()
scores
```

	Accuracy	F1	Recall	Precision
First DF	0.573333	0.288889	0.276596	0.302326

Task 2

Identify Weak Variables by P-Value

Since sklearn doesn't have a way to check P-Value, I will use statsmodels to check P Values.

It can be seen that highest P values are V4,V5,V6 and V7 therefore it is the weakest data.

```
model = sm.Logit(y_train, X_train).fit()
model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.544965
         Iterations 6
```

Logit Regression Results

Dep. Variable:	Label	**No. Observations:**	150
Model:	Logit	**Df Residuals:**	142
Method:	MLE	**Df Model:**	7
Date:	Tue, 15 Dec 2020	**Pseudo R-squ.:**	0.1609
Time:	21:12:05	**Log-Likelihood:**	-81.745
converged:	True	**LL-Null:**	-97.423
Covariance Type:	nonrobust	**LLR p-value:**	5.344e-05

	coef	std err	z	P>|z|	[0.025	0.975]
V1	0.1589	0.067	2.378	0.017	0.028	0.290
V2	0.0244	0.007	3.341	0.001	0.010	0.039
V3	-0.0653	0.015	-4.367	0.000	-0.095	-0.036
V4	-0.0164	0.016	-1.021	0.307	-0.048	0.015
V5	-0.0026	0.002	-1.080	0.280	-0.007	0.002
V6	0.0284	0.028	1.008	0.313	-0.027	0.083
V7	0.0525	0.193	0.272	0.785	-0.325	0.430

Using Backward Elimination

```
step.forwardSelection(X_train, y_train)
```

```
         Current function value: 0.490797
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.496810
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.489531
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.496700
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.496537
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.481943
         Iterations 6
Entered : V1    AIC : 152.58304469923124
Optimization terminated successfully.
         Current function value: 0.473620
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.481900
         Iterations 6

Optimization terminated successfully.
         Current function value: 0.474002
         Iterations 6
Optimization terminated successfully.
```

```
        Current function value: 0.481252
        Iterations 6
Optimization terminated successfully.
         Current function value: 0.481943
         Iterations 6
Break : Significance Level
Optimization terminated successfully.
         Current function value: 0.481943
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Label   No. Observations:                  150
Model:                          Logit   Df Residuals:                      146
Method:                           MLE   Df Model:                            3
Date:                Tue, 15 Dec 2020   Pseudo R-squ.:                  0.2580
Time:                        21:21:11   Log-Likelihood:                -72.292
converged:                       True   LL-Null:                       -97.423
Covariance Type:            nonrobust   LLR p-value:                 7.024e-11
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -6.4735      1.070     -6.051      0.000      -8.570      -4.377
V2             0.0451      0.008      5.327      0.000       0.029       0.062
V5            -0.0046      0.002     -2.229      0.026      -0.009      -0.001
V1             0.1354      0.066      2.061      0.039       0.007       0.264
==============================================================================
AIC: 152.58304469923124
BIC: 164.62558587561625
Final Variables: ['intercept', 'V2', 'V5', 'V1']
(['intercept', 'V2', 'V5', 'V1'],
 '\nEntered : V2\n\n\n                           Logit Regression Results
```

After the backward elimination, we came to conclusion that our final variables will be V2, V5 and V1.

```
new_x = X[['V1','V2','V5']]
new_x.head()
```

	V1	V2	V5
0	1.221400	128.101200	180.956968
1	2.609743	85.891549	52.545356
2	2.682163	99.782456	71.578043
3	3.196969	115.189168	0.269863
4	4.790932	144.487763	-0.033570

Generating New Model

We normally split data by 0.8 for train and 0.2 for test, but I will assume that it is required that we

```
X_train, X_test, y_train, y_test = train_test_split(new_x, y, test_size = 0.5, stratify = y)
```

```
new_model = regression.fit(X_train, y_train)
y_pred_new = model.predict(X_test)
```

New Scores

Here we calculate scores and append the new scores into our dataframe. 0th row is first and 1st row is second logistic model results.

```
new_scores = {}
new_scores['Accuracy'] = accuracy_score(y_pred_new, y_pred)
new_scores['F1'] = f1_score(y_pred_new, y_pred)
new_scores['Recall'] = recall_score(y_pred_new, y_pred)
new_scores['Precision'] = precision_score(y_pred_new, y_pred)

all_scores = scores.append(new_scores, ignore_index=True)
all_scores
```

	Accuracy	F1	Recall	Precision
0	0.573333	0.288889	0.276596	0.302326
1	0.600000	0.318182	0.311111	0.325581

In conclusion, we can see our accuracy, F1, Recall and Precision has increasd slightly after doing a backward step elimination. This shows that our new model has increased performance and accuracy now.