zib43yjdm

September 5, 2023

##Name - Indranil Bain ##Roll No. - 2020CSB039 ###Assignment No - 03 (Titanic)

Download Titanic Dataset(https://www.kaggle.com/heptapod/titanic/version/1#) and do initial pre-processing and train a Logistic Regression for the classifier.

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
BASE_PATH = '/content/drive/MyDrive/CSV Files - COLAB/train_and_test2.csv'
```

[3]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn. preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn. tree import DecisionTreeClassifier
import matplotlib. pyplot as plt
```

[4]:
```
dataset = pd.read_csv(BASE_PATH)
dataset
```

[4]:
```
      Passengerid  Age      Fare  Sex  sibsp  zero  zero.1  zero.2  zero.3  \
0               1  22.0    7.2500    0      1     0       0       0       0
1               2  38.0   71.2833    1      1     0       0       0       0
2               3  26.0    7.9250    1      0     0       0       0       0
3               4  35.0   53.1000    1      1     0       0       0       0
4               5  35.0    8.0500    0      0     0       0       0       0
…             …   …         …   …     …    …      …      …     …
1304         1305  28.0    8.0500    0      0     0       0       0       0
1305         1306  39.0  108.9000    1      0     0       0       0       0
1306         1307  38.5    7.2500    0      0     0       0       0       0
1307         1308  28.0    8.0500    0      0     0       0       0       0
1308         1309  28.0   22.3583    0      1     0       0       0       0

      zero.4  …  zero.12  zero.13  zero.14  Pclass  zero.15  zero.16  \
0          0  …        0        0        0       3        0        0
1          0  …        0        0        0       1        0        0
```

```
2          0  …        0        0        0       3        0        0
3          0  …        0        0        0       1        0        0
4          0  …        0        0        0       3        0        0
…       …  …      …      …      …     …      …      …
1304       0  …        0        0        0       3        0        0
1305       0  …        0        0        0       1        0        0
1306       0  …        0        0        0       3        0        0
1307       0  …        0        0        0       3        0        0
1308       0  …        0        0        0       3        0        0

      Embarked  zero.17  zero.18  2urvived
0          2.0        0        0         0
1          0.0        0        0         1
2          2.0        0        0         1
3          2.0        0        0         1
4          2.0        0        0         0
…          …        …        …        …
1304       2.0        0        0         0
1305       0.0        0        0         0
1306       2.0        0        0         0
1307       2.0        0        0         0
1308       0.0        0        0         0

[1309 rows x 28 columns]
```

[5]:
```
dataset.dropna()
```

[5]:
```
      Passengerid   Age      Fare  Sex  sibsp  zero  zero.1  zero.2  zero.3  \
0               1  22.0    7.2500    0      1     0       0       0       0
1               2  38.0   71.2833    1      1     0       0       0       0
2               3  26.0    7.9250    1      0     0       0       0       0
3               4  35.0   53.1000    1      1     0       0       0       0
4               5  35.0    8.0500    0      0     0       0       0       0
…            …    …        …   …     …    …     …     …     …
1304         1305  28.0    8.0500    0      0     0       0       0       0
1305         1306  39.0  108.9000    1      0     0       0       0       0
1306         1307  38.5    7.2500    0      0     0       0       0       0
1307         1308  28.0    8.0500    0      0     0       0       0       0
1308         1309  28.0   22.3583    0      1     0       0       0       0

      zero.4  …  zero.12  zero.13  zero.14  Pclass  zero.15  zero.16  \
0          0  …        0        0        0       3        0        0
1          0  …        0        0        0       1        0        0
2          0  …        0        0        0       3        0        0
3          0  …        0        0        0       1        0        0
4          0  …        0        0        0       3        0        0
…       …  …      …      …      …     …      …      …
```

```
1304       0  …       0       0       0      3       0       0
1305       0  …       0       0       0      1       0       0
1306       0  …       0       0       0      3       0       0
1307       0  …       0       0       0      3       0       0
1308       0  …       0       0       0      3       0       0

      Embarked  zero.17  zero.18  2urvived
0          2.0        0        0         0
1          0.0        0        0         1
2          2.0        0        0         1
3          2.0        0        0         1
4          2.0        0        0         0
…          …       …       …       …
1304       2.0        0        0         0
1305       0.0        0        0         0
1306       2.0        0        0         0
1307       2.0        0        0         0
1308       0.0        0        0         0

[1307 rows x 28 columns]
```

```
[6]: dataset.columns
```

```
[6]: Index(['Passengerid', 'Age', 'Fare', 'Sex', 'sibsp', 'zero', 'zero.1',
            'zero.2', 'zero.3', 'zero.4', 'zero.5', 'zero.6', 'Parch', 'zero.7',
            'zero.8', 'zero.9', 'zero.10', 'zero.11', 'zero.12', 'zero.13',
            'zero.14', 'Pclass', 'zero.15', 'zero.16', 'Embarked', 'zero.17',
            'zero.18', '2urvived'],
           dtype='object')
```

```
[7]: dataset = dataset[
         filter(
     lambda colName: "zero" not in colName,
             dataset.columns
         )
     ]
     dataset = dataset.drop("Passengerid", axis=1)
     dataset
```

```
[7]:       Age     Fare  Sex  sibsp  Parch  Pclass  Embarked  2urvived
     0     22.0   7.2500    0      1      0       3       2.0         0
     1     38.0  71.2833    1      1      0       1       0.0         1
     2     26.0   7.9250    1      0      0       3       2.0         1
     3     35.0  53.1000    1      1      0       1       2.0         1
     4     35.0   8.0500    0      0      0       3       2.0         0
     …     …       …   …    …     …      …       …        …
     1304  28.0   8.0500    0      0      0       3       2.0         0
```

```
1305  39.0  108.9000    1      0      0       1       0.0         0
1306  38.5    7.2500    0      0      0       3       2.0         0
1307  28.0    8.0500    0      0      0       3       2.0         0
1308  28.0   22.3583    0      1      1       3       0.0         0

[1309 rows x 8 columns]
```

[14]:
```python
from sklearn.preprocessing import OneHotEncoder

def one_hot_encode(X: "pd.DataFrame", col_name: "str") -> "pd.DataFrame":
    encoder = OneHotEncoder()
    encoded_df = pd.DataFrame(
        encoder.fit_transform(X[[col_name]]).toarray(),
        index=X.index,
        columns=encoder.get_feature_names_out()
    )
    X = X.join(encoded_df)
    X = X.drop(col_name, axis=1)

    return X
```

[15]:
```python
columns_to_encode = ["Pclass", "Embarked", "Sex"]

for column in columns_to_encode:
    dataset = one_hot_encode(dataset, column)

dataset
```

[15]:
```
       Age      Fare  sibsp  Parch  2urvived  Pclass_1  Pclass_2  Pclass_3  \
0     22.0    7.2500      1      0         0       0.0       0.0       1.0
1     38.0   71.2833      1      0         1       1.0       0.0       0.0
2     26.0    7.9250      0      0         1       0.0       0.0       1.0
3     35.0   53.1000      1      0         1       1.0       0.0       0.0
4     35.0    8.0500      0      0         0       0.0       0.0       1.0
…    …       …    …    …       …       …       …       …
1304  28.0    8.0500      0      0         0       0.0       0.0       1.0
1305  39.0  108.9000      0      0         0       1.0       0.0       0.0
1306  38.5    7.2500      0      0         0       0.0       0.0       1.0
1307  28.0    8.0500      0      0         0       0.0       0.0       1.0
1308  28.0   22.3583      1      1         0       0.0       0.0       1.0

      Embarked_0.0  Embarked_1.0  Embarked_2.0  Embarked_nan  Sex_0  Sex_1
0              0.0           0.0           1.0           0.0    1.0    0.0
1              1.0           0.0           0.0           0.0    0.0    1.0
2              0.0           0.0           1.0           0.0    0.0    1.0
3              0.0           0.0           1.0           0.0    0.0    1.0
4              0.0           0.0           1.0           0.0    1.0    0.0
```

```
…            …           …           …           …    …    …
1304           0.0           0.0           1.0           0.0    1.0    0.0
1305           1.0           0.0           0.0           0.0    0.0    1.0
1306           0.0           0.0           1.0           0.0    1.0    0.0
1307           0.0           0.0           1.0           0.0    1.0    0.0
1308           1.0           0.0           0.0           0.0    1.0    0.0

[1309 rows x 14 columns]
```

[19]:

```python
# Age and Fare needs to be standardized
from sklearn.preprocessing import StandardScaler
def standardize(df: "pd.DataFrame", col_name: "str") -> "pd.DataFrame":

    scaler = StandardScaler()

    df[[col_name]] = pd.DataFrame(
        data=scaler.fit_transform(df[[col_name]]),
        index=df.index,
        columns=[col_name]
    )
    return df
```

[20]:

```python
columns_to_standardize = ['Age', "Fare", 'sibsp', "Parch"]

for column in columns_to_standardize:
    dataset = standardize(dataset, column)

dataset
```

[20]:

```
           Age      Fare     sibsp     Parch  2urvived  Pclass_1  Pclass_2  \
0    -0.581628 -0.503291  0.481288 -0.445000         0       0.0       0.0
1     0.658652  0.734744  0.481288 -0.445000         1       1.0       0.0
2    -0.271558 -0.490240 -0.479087 -0.445000         1       0.0       0.0
3     0.426099  0.383183  0.481288 -0.445000         1       1.0       0.0
4     0.426099 -0.487824 -0.479087 -0.445000         0       0.0       0.0
…        …       …       …       …       …       …       …
1304 -0.116523 -0.487824 -0.479087 -0.445000         0       0.0       0.0
1305  0.736169  1.462034 -0.479087 -0.445000         0       1.0       0.0
1306  0.697411 -0.503291 -0.479087 -0.445000         0       0.0       0.0
1307 -0.116523 -0.487824 -0.479087 -0.445000         0       0.0       0.0
1308 -0.116523 -0.211184  0.481288  0.710763         0       0.0       0.0

      Pclass_3  Embarked_0.0  Embarked_1.0  Embarked_2.0  Embarked_nan  Sex_0  \
0          1.0           0.0           0.0           1.0           0.0    1.0
1          0.0           1.0           0.0           0.0           0.0    0.0
2          1.0           0.0           0.0           1.0           0.0    0.0
3          0.0           0.0           0.0           1.0           0.0    0.0
```

```
4         1.0          0.0          0.0          1.0          0.0    1.0
…       …          …          …          …          …       …
1304      1.0          0.0          0.0          1.0          0.0    1.0
1305      0.0          1.0          0.0          0.0          0.0    0.0
1306      1.0          0.0          0.0          1.0          0.0    1.0
1307      1.0          0.0          0.0          1.0          0.0    1.0
1308      1.0          1.0          0.0          0.0          0.0    1.0

      Sex_1
0       0.0
1       1.0
2       1.0
3       1.0
4       0.0
…     …
1304    0.0
1305    1.0
1306    0.0
1307    0.0
1308    0.0

[1309 rows x 14 columns]
```

[22]:

```python
# Preprocessing Done, lets move to model
X = dataset.drop('2urvived', axis=1)
y = dataset[['2urvived']]
```

[23]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

[25]:

```python
# make, train, and score the model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(X_train, y_train.iloc[:,0])
accuracy = model.score(X_test, y_test)
print(f"accuracy = {accuracy}")
```

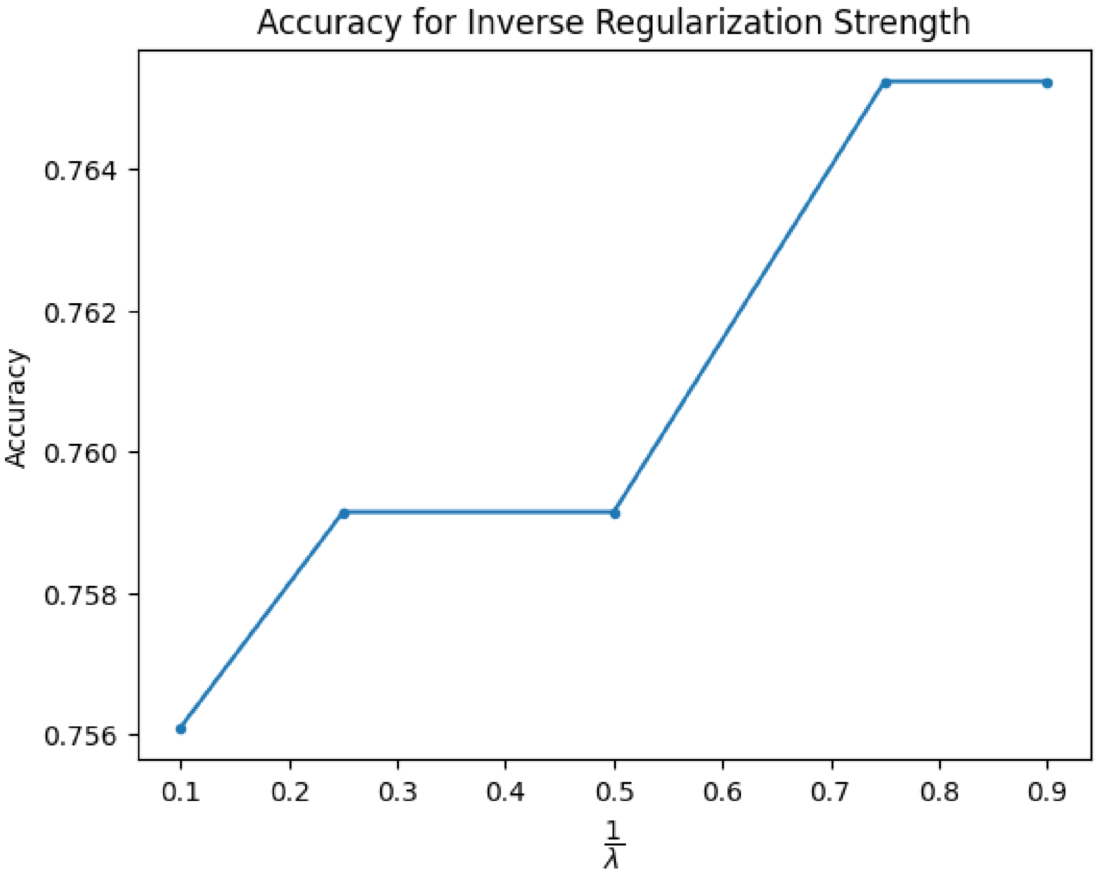
```
accuracy = 0.7652439024390244
```

2. Analyze and control the overfitting by varying the inverse of regularization strength parameter (0.1, 0.25,0.5, 0.75, 0.9) and plot the accuracy graph for the test set.

[26]:

```python
import matplotlib.pyplot as plt

def get_acc_log_reg( X_train: "pd.DataFrame", X_test: "pd.DataFrame", y_train:␣
 ↪"pd.DataFrame", y_test: "pd.DataFrame", c=1.0
) -> "float":
      return LogisticRegression(C=c)\
      .fit(X_train, y_train.iloc[:, 0])\
      .score(X_test, y_test)
inv_reg_strs = (0.1, 0.25, 0.5, 0.75, 0.9)
accuracies = [get_acc_log_reg(X_train, X_test, y_train, y_test, c) for c in␣
 ↪inv_reg_strs]
plt.plot(inv_reg_strs, accuracies, '.-')
plt.title("Accuracy for Inverse Regularization Strength")
plt.xlabel(r"$\dfrac{1}{\lambda}$")
plt.ylabel("Accuracy")
plt.show()
```



[28]:
```
pd.DataFrame(
    data = zip(inv_reg_strs, accuracies),
    columns=['inv_reg_str', 'accuracy']
)
```

[28]:
```
   inv_reg_str  accuracy
0         0.10  0.756098
1         0.25  0.759146
2         0.50  0.759146
```

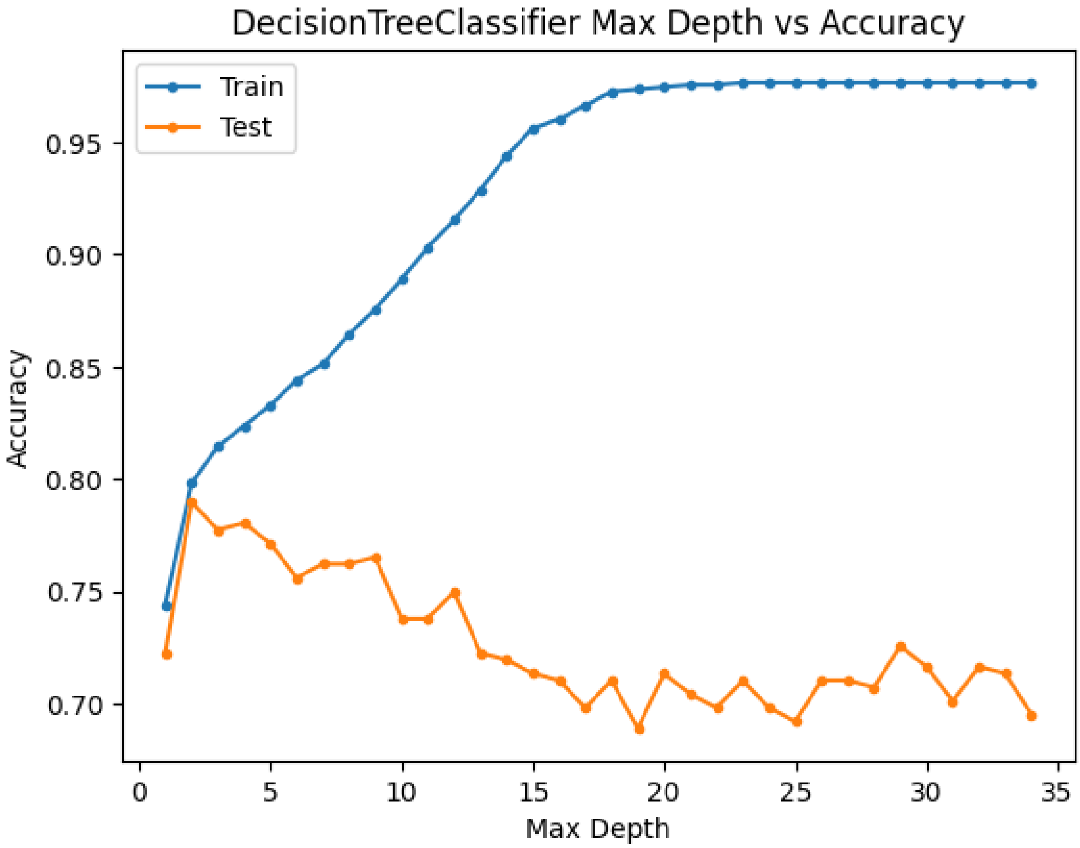
```
3          0.75  0.765244
4          0.90  0.765244
```

3. Using the same dataset train a Decision Tree classifier and vary the maximum depth of the tree to train at least 5 classifiers to analyze the effectiveness.

[30]:

```
from sklearn.tree import DecisionTreeClassifier
def get_acc_dec_tree(
    X_train: "pd.DataFrame",
    X_test: "pd.DataFrame",
    y_train: "pd.DataFrame",
    y_test: "pd.DataFrame",
    max_depth=1
) -> "float":
      return DecisionTreeClassifier(max_depth=max_depth)\
        .fit(X_train, y_train)\
        .score(X_test, y_test)
max_depths = range(1, 35)
train_accuracies = [get_acc_dec_tree(X_train, X_train, y_train, y_train, max_d)␣
 ↪for max_d in max_depths]
test_accuracies = [get_acc_dec_tree(X_train, X_test, y_train, y_test, max_d)␣
 ↪for max_d in max_depths]


plt.plot(max_depths, train_accuracies, ".-", label='Train')
plt.plot(max_depths, test_accuracies, ".-", label='Test')
plt.title("DecisionTreeClassifier Max Depth vs Accuracy")
plt.xlabel("Max Depth")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```


DecisionTreeClassifier Max Depth vs Accuracy
Train
Test
Accuracy
0.70
0.75
0.80
0.85
0.90
0.95
0
5
10
15
20
25
30
35
Max Depth