BaselineModelTraining

March 8, 2024

1 Milestone 1: Baseline Model Training

In this file we simply converted our categorical values into integers and trained a logistic regression model and a decision tree model to serve as baseline models.

[34]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import roc_curve, roc_auc_score
import pandas as pd
```

[35]:
```
df = pd.read_csv("Assign1Data.csv")
df.head()
```

[35]:
```
   Gender Senior Citizen Partner  Tenure Months Phone Service Multiple Lines  \
0    Male             No      No              2           Yes             No
1  Female             No      No              2           Yes             No
2  Female             No      No              8           Yes            Yes
3  Female             No     Yes             28           Yes            Yes
4    Male             No      No             49           Yes            Yes

  Internet Service Online Security Online Backup Device Protection  \
0              DSL             Yes           Yes                No
1      Fiber optic              No            No                No
2      Fiber optic              No            No               Yes
3      Fiber optic              No            No               Yes
4      Fiber optic              No           Yes               Yes

  Tech Support Streaming TV Streaming Movies        Contract  \
0           No           No               No  Month-to-month
1           No           No               No  Month-to-month
2           No          Yes              Yes  Month-to-month
3          Yes          Yes              Yes  Month-to-month
```

```
4           No          Yes              Yes  Month-to-month

  Paperless Billing             Payment Method  Monthly Charges  \
0               Yes               Mailed check            53.85
1               Yes           Electronic check            70.70
2               Yes           Electronic check            99.65
3               Yes           Electronic check           104.80
4               Yes  Bank transfer (automatic)           103.70

   Total Charges  Churn Value
0         108.15            1
1         151.65            1
2         820.50            1
3        3046.05            1
4        5036.30            1
```

[36]:

```python
# Converting the categorical values into integers to train our models

for col in df.select_dtypes(include=['object']).columns:
    df[col] = df[col].astype('category').cat.codes

df.head()
```

[36]:

```
   Gender  Senior Citizen  Partner  Tenure Months  Phone Service  \
0       1               0        0              2              1
1       0               0        0              2              1
2       0               0        0              8              1
3       0               0        1             28              1
4       1               0        0             49              1

   Multiple Lines  Internet Service  Online Security  Online Backup  \
0               0                 0                2              2
1               0                 1                0              0
2               2                 1                0              0
3               2                 1                0              0
4               2                 1                0              2

   Device Protection  Tech Support  Streaming TV  Streaming Movies  Contract  \
0                  0             0             0                 0         0
1                  0             0             0                 0         0
2                  2             0             2                 2         0
3                  2             2             2                 2         0
4                  2             0             2                 2         0

   Paperless Billing  Payment Method  Monthly Charges  Total Charges  \
0                  1               3            53.85         108.15
1                  1               2            70.70         151.65
```

```
2                 1              2         99.65         820.50
3                 1              2        104.80        3046.05
4                 1              0        103.70        5036.30

   Churn Value
0            1
1            1
2            1
3            1
4            1
```

[37]:
```python
df.describe()
```

[37]:
```
            Gender  Senior Citizen      Partner  Tenure Months  Phone Service  \
count  7043.000000     7043.000000  7043.000000    7043.000000    7043.000000
mean      0.504756        0.162147     0.483033      32.371149       0.903166
std       0.500013        0.368612     0.499748      24.559481       0.295752
min       0.000000        0.000000     0.000000       0.000000       0.000000
25%       0.000000        0.000000     0.000000       9.000000       1.000000
50%       1.000000        0.000000     0.000000      29.000000       1.000000
75%       1.000000        0.000000     1.000000      55.000000       1.000000
max       1.000000        1.000000     1.000000      72.000000       1.000000

       Multiple Lines  Internet Service  Online Security  Online Backup  \
count     7043.000000       7043.000000      7043.000000    7043.000000
mean         0.940508          0.872923         0.790004       0.906432
std          0.948554          0.737796         0.859848       0.880162
min          0.000000          0.000000         0.000000       0.000000
25%          0.000000          0.000000         0.000000       0.000000
50%          1.000000          1.000000         1.000000       1.000000
75%          2.000000          1.000000         2.000000       2.000000
max          2.000000          2.000000         2.000000       2.000000

       Device Protection  Tech Support  Streaming TV  Streaming Movies  \
count        7043.000000   7043.000000   7043.000000       7043.000000
mean            0.904444      0.797104      0.985376          0.992475
std             0.879949      0.861551      0.885002          0.885091
min             0.000000      0.000000      0.000000          0.000000
25%             0.000000      0.000000      0.000000          0.000000
50%             1.000000      1.000000      1.000000          1.000000
75%             2.000000      2.000000      2.000000          2.000000
max             2.000000      2.000000      2.000000          2.000000

          Contract  Paperless Billing  Payment Method  Monthly Charges  \
count  7043.000000        7043.000000     7043.000000      7043.000000
mean      0.690473           0.592219        1.574329        64.761692
std       0.833755           0.491457        1.068104        30.090047
```

```
min       0.000000          0.000000        0.000000        18.250000
25%       0.000000          0.000000        1.000000        35.500000
50%       0.000000          1.000000        2.000000        70.350000
75%       1.000000          1.000000        2.000000        89.850000
max       2.000000          1.000000        3.000000       118.750000

       Total Charges  Churn Value
count    7043.000000  7043.000000
mean     2283.300441     0.265370
std      2265.000258     0.441561
min        18.800000     0.000000
25%       402.225000     0.000000
50%      1400.550000     0.000000
75%      3786.600000     1.000000
max      8684.800000     1.000000
```

[38]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             7043 non-null   int8
 1   Senior Citizen     7043 non-null   int8
 2   Partner            7043 non-null   int8
 3   Tenure Months      7043 non-null   int64
 4   Phone Service      7043 non-null   int8
 5   Multiple Lines     7043 non-null   int8
 6   Internet Service   7043 non-null   int8
 7   Online Security    7043 non-null   int8
 8   Online Backup      7043 non-null   int8
 9   Device Protection  7043 non-null   int8
 10  Tech Support       7043 non-null   int8
 11  Streaming TV       7043 non-null   int8
 12  Streaming Movies   7043 non-null   int8
 13  Contract           7043 non-null   int8
 14  Paperless Billing  7043 non-null   int8
 15  Payment Method     7043 non-null   int8
 16  Monthly Charges    7043 non-null   float64
 17  Total Charges      7043 non-null   float64
 18  Churn Value        7043 non-null   int64
dtypes: float64(2), int64(2), int8(15)
memory usage: 323.4 KB
```

[39]:

```python
X = df.drop(['Churn Value'], axis=1)
y = df['Churn Value']
```

```python
X_train, X_test_final, y_train, y_test_final = train_test_split(X, y,␣
 ↪test_size=0.2, random_state=42)

# printing for varification
print("data shapes:")
print("X_train shape:", X_train.shape)
print("X_test_final shape:", X_test_final.shape)
print("y_train shape:", y_train.shape)
print("y_test_final shape:", y_test_final.shape)

X_train_val, X_test_validation, y_train_val, y_test_validation =␣
 ↪train_test_split(X_train, y_train, test_size=0.2, random_state=42)
print("X_train_val shape:", X_train_val.shape)
print("X_test_validation shape:", X_test_validation.shape)
print("y_train_val shape:", y_train_val.shape)
print("y_test_validation shape:", y_test_validation.shape)
```

```
data shapes:
X_train shape: (5634, 18)
X_test_final shape: (1409, 18)
y_train shape: (5634,)
y_test_final shape: (1409,)
X_train_val shape: (4507, 18)
X_test_validation shape: (1127, 18)
y_train_val shape: (4507,)
y_test_validation shape: (1127,)
```

Here we split our data into 3 parts. First we divided the whole data set into 80/20, kept the 20 percent data aside for final testing. The 80 percent data was then used to get another 20% data from within it as validation set.

Moving forward we trained our logistic regression and decision tree classifier with the training data and check their prediction on the validation dataset

[40]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score
from sklearn.metrics import classification_report

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_val, y_train_val)

dec_tree = DecisionTreeClassifier(random_state=42)
dec_tree.fit(X_train_val, y_train_val)

y_pred_log_reg = log_reg.predict(X_test_validation)
y_pred_dec_tree = dec_tree.predict(X_test_validation)

print("Logistic Regression")
```

```python
print(classification_report(y_test_validation, y_pred_log_reg))

print ("")

print("Decision Tree")
print(classification_report(y_test_validation, y_pred_dec_tree))
```

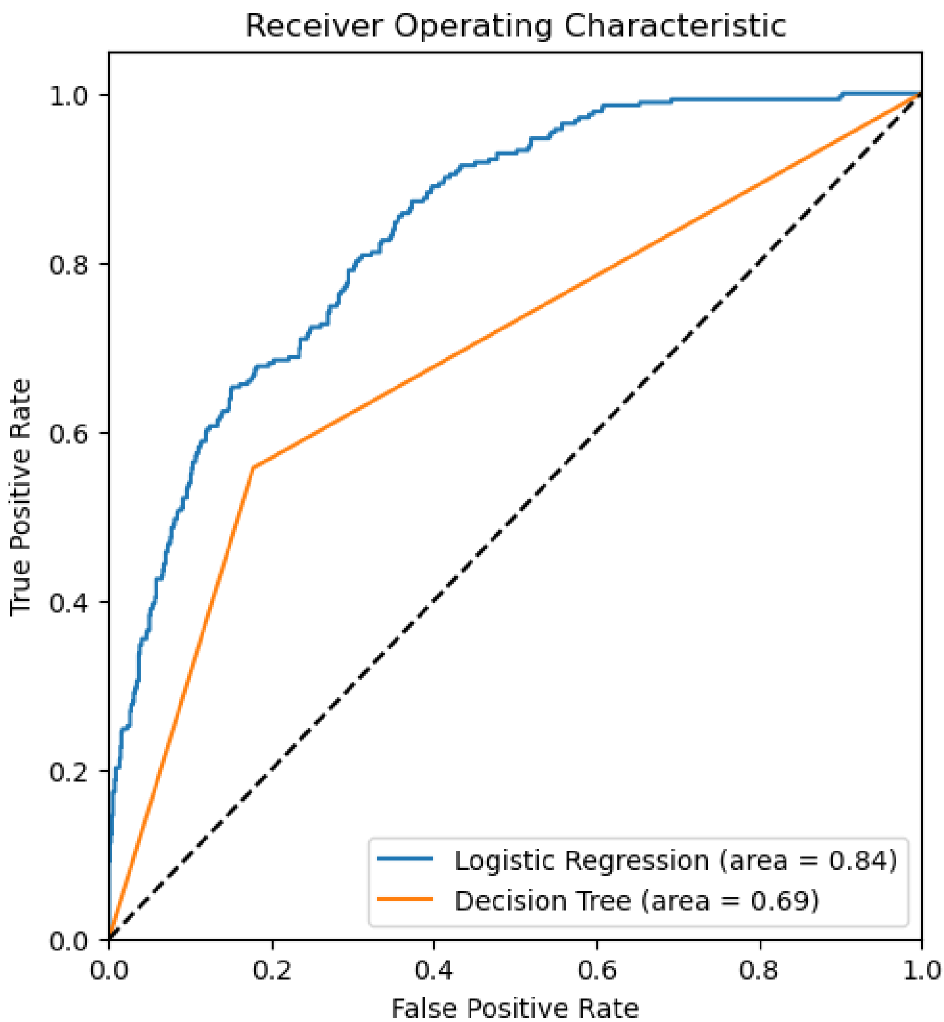
```
Logistic Regression
              precision    recall  f1-score   support

           0       0.86      0.89      0.88       845
           1       0.64      0.56      0.60       282

    accuracy                           0.81      1127
   macro avg       0.75      0.73      0.74      1127
weighted avg       0.81      0.81      0.81      1127


Decision Tree
              precision    recall  f1-score   support

           0       0.85      0.82      0.84       845
           1       0.51      0.55      0.53       282

    accuracy                           0.76      1127
   macro avg       0.68      0.69      0.68      1127
weighted avg       0.76      0.76      0.76      1127
```

We used classification_report() to build a text report showing the main classification metrics (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html). The metrices are to evaluate our baseline model training. We also printed the AUROC curve to visually show the TPR and FPR correlation.

[41]:
```python
from sklearn.metrics import roc_curve, auc, precision_recall_curve
import matplotlib.pyplot as plt

y_scores_lr = log_reg.predict_proba(X_test_validation)[:, 1]
y_scores_dt = dec_tree.predict_proba(X_test_validation)[:, 1]

fpr_lr, tpr_lr, _ = roc_curve(y_test_validation, y_scores_lr)
roc_auc_lr = auc(fpr_lr, tpr_lr)

fpr_dt, tpr_dt, _ = roc_curve(y_test_validation, y_scores_dt)
roc_auc_dt = auc(fpr_dt, tpr_dt)

# Plot ROC curve
plt.figure(figsize=(12, 6))
```

```
plt.subplot(1, 2, 1)
plt.plot(fpr_lr, tpr_lr, label='Logistic Regression (area = %0.2f)' %␣
↪roc_auc_lr)
plt.plot(fpr_dt, tpr_dt, label='Decision Tree (area = %0.2f)' % roc_auc_dt)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

[41]: <matplotlib.legend.Legend at 0x7fb3a6253e20>



Below is the code to find resource consumption for training the two models followed

by the output

[42]:
```python
import pandas as pd
import time
import memory_profiler
from sklearn import model_selection as skms, linear_model, tree, svm, ensemble

def lr_go(train_ftrs, train_tgt):
    linreg = linear_model.LinearRegression()
    return linreg.fit(train_ftrs, train_tgt)

def dt_go(train_ftrs, train_tgt):
    dtree = tree.DecisionTreeClassifier()
    return dtree.fit(train_ftrs, train_tgt)

# Function to measure time and memory
def measure_model(model_func, train_ftrs, train_tgt):
    start_time = time.time()
    mem_usage_start = memory_profiler.memory_usage(max_usage=True)
    model = model_func(train_ftrs, train_tgt)
    mem_usage_end = memory_profiler.memory_usage(max_usage=True)
    end_time = time.time()

    print(f"{model_func.__name__}:")
    print(f" Time: {end_time - start_time:.4f} seconds")
    print(f" Memory: {mem_usage_end - mem_usage_start:.4f} MiB\n")


# Measure each model
for model_func in [lr_go, dt_go]:
    measure_model(model_func, X_train_val, y_train_val)
```

```
lr_go:
 Time: 0.2075 seconds
 Memory: 0.0039 MiB

dt_go:
 Time: 0.2251 seconds
 Memory: 0.3242 MiB
```

1.1 The following part below was done after the milestone 3 was completed to check the performance of the baseline models on the final test data

[43]:
```python
from sklearn.metrics import classification_report

y_pred_log_reg = log_reg.predict(X_test_final)
y_pred_dec_tree = dec_tree.predict(X_test_final)
```

```python
print("Logistic Regression")
print(classification_report(y_test_final, y_pred_log_reg))

print ("")

print("Decision Tree")
print(classification_report(y_test_final, y_pred_dec_tree))
```

```
Logistic Regression
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1009
           1       0.66      0.56      0.61       400

    accuracy                           0.79      1409
   macro avg       0.75      0.72      0.73      1409
weighted avg       0.79      0.79      0.79      1409


Decision Tree
              precision    recall  f1-score   support

           0       0.81      0.82      0.81      1009
           1       0.53      0.53      0.53       400

    accuracy                           0.73      1409
   macro avg       0.67      0.67      0.67      1409
weighted avg       0.73      0.73      0.73      1409
```