Week5_Assignment

February 20, 2024

0.1 Import Libraries and modules

[1]:
```
import pandas as pd
import pickle
from IPython.display import Code
from pycaret.classification import setup, compare_models, predict_model,␣
↪save_model, load_model
import numpy as np
```

[2]:
```
df = pd.read_csv("cleaned_churn_data.csv")
df.tail(5)
```

[2]:
```
      tenure  PhoneService  Contract  MonthlyCharges  TotalCharges  Churn  \
7038      24             1         1           84.80       1990.50      0
7039      72             1         1          103.20       7362.90      0
7040      11             0         0           29.60        346.45      0
7041       4             1         0           74.40        306.60      1
7042      66             1         3          105.65       6844.50      0

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
7038                        3.533333                          0
7039                        1.433333                          0
7040                        2.690909                          0
7041                       18.600000                          0
7042                        1.600758                          1

      Credit card (automatic)  Electronic check  Mailed check
7038                        0                 1             1
7039                        1                 1             0
7040                        0                 0             0
7041                        0                 1             1
7042                        0                 1             0
```

0.2 Handle Infinity values in the dataset

[3]:
```
columns_with_infinity = df.columns[np.isinf(df).any()]

print("Columns with Infinity values:", columns_with_infinity)

# Replace infinity values
df[columns_with_infinity] = df[columns_with_infinity].replace([np.inf, -np.
 ↪inf], np.nan)
```

```
Columns with Infinity values: Index(['MonthlyCharges_to_tenure_Ratio'],
dtype='object')
```

The column `MonthlyCharges_to_tenure_Ratio`, in the DataFrame (df) contains infinity values. We've identified and printed it, and replaced these infinity values with NaN.

0.3 auto ML environment

[4]:
```
automl = setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x7fa313cfcad0>
```

This output summarizes the setup information for the PyCaret auto ML environment designed for the binary classification task predicting `Churn`.

The key points include,

The session has an ID of 6993, and the target variable is Churn, categorized as binary. The original dataset has dimensions (7043, 11), and after transformation, it maintains the same shape. The transformed training set comprises 4930 samples, while the transformed test set has 2113 samples. There are 10 numeric features in the dataset, and the percentage of rows with missing values is 0.2%.

The data has undergone preprocessing with simple imputation. Numeric features have been imputed with the mean, and categorical features with the mode. The cross-validation is performed using StratifiedKFold with 10 folds. The setup utilizes all available CPUs (-1 CPU jobs) and does not employ GPU acceleration. Logging of the experiment is turned off, and the experiment is named `clf-default-name` with a unique session identifier (USI) of df3d.

The dataset is well-prepared, and the setup is ready for model comparison and selection.

[5]:
```
# Access the elements of the automl_setup object
automl_element = automl.get_config("X_train")
automl_element
```

[5]:
```
      tenure  PhoneService  Contract  MonthlyCharges  TotalCharges  \
1086      11             1         0       89.699997   1047.699951
6871      52             1         0       94.599998   5025.799805
669       70             0         3       57.799999   4039.300049
3864      30             1         0      100.199997   2983.800049
3364      21             1         0      103.900002   2254.199951
…        …           …        …              …             …
```

```
2889      58              0          1       50.000000   2919.850098
1046      52              1          1       74.000000   3877.649902
2969      65              1          3      109.300003   7337.549805
572       11              1          1       64.900002    697.250000
1276      71              1          3       99.400002   7168.250000

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
1086                        8.154546                          1
6871                        1.819231                          1
669                         0.825714                          1
3864                        3.340000                          0
3364                        4.947619                          0
…                              …                         …
2889                        0.862069                          0
1046                        1.423077                          0
2969                        1.681538                          0
572                         5.900000                          0
1276                        1.400000                          0

      Credit card (automatic)  Electronic check  Mailed check
1086                        0                 1             0
6871                        0                 1             0
669                         0                 1             0
3864                        0                 0             0
3364                        0                 0             0
…                        …                …            …
2889                        0                 1             1
1046                        1                 1             0
2969                        0                 0             0
572                         1                 1             0
1276                        0                 0             0

[4930 rows x 10 columns]
```

0.4 Compare classification models

[6]:
```python
best_model = compare_models()
```

```
<IPython.core.display.HTML object>

<pandas.io.formats.style.Styler at 0x7fa31345d1d0>

<IPython.core.display.HTML object>
```

he output includes information about the PyCaret setup and the performance of various classification models.

The Linear Discriminant Analysis (LDA) model has the highest accuracy among the models listed.

`LDA (Linear Discriminant Analysis):`

```
Accuracy: 0.7929
AUC: 0.8346
Recall: 0.4357
Precision: 0.6683
F1 Score: 0.5265
Kappa: 0.4017
MCC: 0.4172
Training Time (Sec): 0.2050
```

LDA demonstrates a good balance between accuracy, precision, recall, and F1 score.

[7]:
```
best_model
```

[7]:
```
LinearDiscriminantAnalysis(covariance_estimator=None, n_components=None,
                           priors=None, shrinkage=None, solver='svd',
                           store_covariance=False, tol=0.0001)
```

0.5 Select rows

[8]:
```
rows = df.iloc[7000:7010]
rows
```

[8]:
```
      tenure  PhoneService  Contract  MonthlyCharges  TotalCharges  Churn  \
7000      67             1         1           20.55       1343.40      0
7001       3             1         0           49.90        130.10      1
7002      64             1         3          105.40       6794.75      0
7003      26             0         0           35.75       1022.50      0
7004      38             1         0           95.10       3691.20      0
7005      23             1         1           19.30        486.20      0
7006      40             1         0          104.50       4036.85      1
7007      72             0         3           63.10       4685.55      0
7008       3             1         0           75.05        256.25      1
7009      23             1         0           81.00       1917.10      1

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
7000                        0.306716                          0
7001                       16.633333                          0
7002                        1.646875                          1
7003                        1.375000                          0
7004                        2.502632                          0
7005                        0.839130                          0
7006                        2.612500                          0
7007                        0.876389                          1
7008                       25.016667                          0
7009                        3.521739                          0

      Credit card (automatic)  Electronic check  Mailed check
7000                        0                 0             0
```

```
7001                         0                 1                 1
7002                         0                 1                 0
7003                         0                 0                 0
7004                         1                 1                 0
7005                         1                 1                 0
7006                         1                 1                 0
7007                         0                 1                 0
7008                         1                 1                 0
7009                         0                 0                 0
```

0.6 Use best_model to predict churn for the rows

[9]:
```python
predicted_rows = predict_model(best_model, rows)
predicted_rows
```

```
<pandas.io.formats.style.Styler at 0x7fa368b54750>
```

[9]:
```
      tenure  PhoneService  Contract  MonthlyCharges  TotalCharges  \
7000      67             1         1       20.549999   1343.400024
7001       3             1         0       49.900002    130.100006
7002      64             1         3      105.400002   6794.750000
7003      26             0         0       35.750000   1022.500000
7004      38             1         0       95.099998   3691.199951
7005      23             1         1       19.299999    486.200012
7006      40             1         0      104.500000   4036.850098
7007      72             0         3       63.099998   4685.549805
7008       3             1         0       75.050003    256.250000
7009      23             1         0       81.000000   1917.099976

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
7000                        0.306716                          0
7001                       16.633333                          0
7002                        1.646875                          1
7003                        1.375000                          0
7004                        2.502632                          0
7005                        0.839130                          0
7006                        2.612500                          0
7007                        0.876389                          1
7008                       25.016666                          0
7009                        3.521739                          0

      Credit card (automatic)  Electronic check  Mailed check  Churn  \
7000                        0                 0             0      0
7001                        0                 1             1      1
7002                        0                 1             0      0
7003                        0                 0             0      0
7004                        1                 1             0      0
```

```
7005                          1                 1             0      0
7006                          1                 1             0      1
7007                          0                 1             0      0
7008                          1                 1             0      1
7009                          0                 0             0      1

      prediction_label  prediction_score
7000                 0            0.9274
7001                 0            0.7149
7002                 0            0.9466
7003                 0            0.7444
7004                 0            0.7630
7005                 0            0.9452
7006                 0            0.7288
7007                 0            0.9550
7008                 1            0.5804
7009                 0            0.5804
```

The selected rows are predicted using the Linear Discriminant Analysis (LDA) model.

LDA Model Performance Metrics:

```
Accuracy: 0.7000
AUC: 1.0000
Recall: 0.2500
Precision: 1.0000
F1 Score: 0.4000
Kappa: 0.2857
MCC: 0.408
```

These scores suggest that the model has achieved optimal performance on the selected rows.

Each row shows the model's prediction_label (0 or 1) and prediction_score, indicating the model's confidence in its predictions.

0.7 Save and serialize model

[10]:
```
save_model(best_model, 'LDA')
```

```
Transformation Pipeline and Model Successfully Saved
```

[10]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure', 'PhoneService',
                                              'Contract', 'MonthlyCharges',
                                              'TotalCharges',
                                              'MonthlyCharges_to_tenure_Ratio',
                                              'Bank transfer (automatic)',
                                              'Credit card (automatic)',
```

```
                                                'Electronic check',
                                                'Mailed check'],
transformer=SimpleImputer(add_indicator=False,
                                                                copy=True,
                                                                fill…
strategy='most_frequent',
verbose='deprecated'))),
                 ('clean_column_names',
                  TransformerWrapper(exclude=None, include=None,
transformer=CleanColumnNames(match='[\\]\\[\\,\\{\\}\\"\\:]+'))),
                 ('trained_model',
                  LinearDiscriminantAnalysis(covariance_estimator=None,
                                             n_components=None, priors=None,
                                             shrinkage=None, solver='svd',
                                             store_covariance=False,
                                             tol=0.0001))],
          verbose=False),
 'LDA.pkl')
```

[11]:
```python
with open('LDA_model.pk', 'wb') as f:
    pickle.dump(best_model, f)
```

[12]:
```python
with open('LDA_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

0.8 Create new data and save to csv

[13]:
```python
new_data = rows.copy()
new_data.drop('Churn', axis=1, inplace=True)
new_data.to_csv('new_churn_data.csv', index=False)
```

[14]:
```python
loaded_model_prediction = loaded_model.predict(new_data)
loaded_model_prediction
```

[14]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int8)
```

The loaded_model_prediction array contains the binary predictions for each row in new_data. The values of 0 and 1 indicate the predicted class (churn or non-churn).

0.9 Probability of churn for each new prediction

[15]:
```python
probability_of_churn = loaded_model.predict_proba(new_data)[:, 1]
probability_of_churn
```

[15]:
```
array([0.07255947, 0.28508461, 0.05336652, 0.25559998, 0.23702122,
       0.05475078, 0.27124794, 0.04503105, 0.5804218 , 0.41962034])
```

The probability_of_churn array contains the predicted probabilities of churn for each corresponding row in new_data. These values represent the confidence that the predicted class is 1 (churn).

For example, a probability of 0.5804 suggests a 58.04% likelihood of churn for the ninth row in new_data.

[17]:
```python
loaded_lda = load_model('LDA')
```

```
Transformation Pipeline and Model Successfully Loaded
```

[18]:
```python
loaded_lda_prediction = predict_model(loaded_lda, new_data)
loaded_lda_prediction
```

```
<IPython.core.display.HTML object>
```

[18]:
```
      tenure  PhoneService  Contract  MonthlyCharges  TotalCharges  \
7000      67             1         1       20.549999   1343.400024
7001       3             1         0       49.900002    130.100006
7002      64             1         3      105.400002   6794.750000
7003      26             0         0       35.750000   1022.500000
7004      38             1         0       95.099998   3691.199951
7005      23             1         1       19.299999    486.200012
7006      40             1         0      104.500000   4036.850098
7007      72             0         3       63.099998   4685.549805
7008       3             1         0       75.050003    256.250000
7009      23             1         0       81.000000   1917.099976

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
7000                        0.306716                          0
7001                       16.633333                          0
7002                        1.646875                          1
7003                        1.375000                          0
7004                        2.502632                          0
7005                        0.839130                          0
7006                        2.612500                          0
7007                        0.876389                          1
7008                       25.016666                          0
7009                        3.521739                          0

      Credit card (automatic)  Electronic check  Mailed check  \
7000                        0                 0             0
7001                        0                 1             1
7002                        0                 1             0
7003                        0                 0             0
7004                        1                 1             0
7005                        1                 1             0
7006                        1                 1             0
7007                        0                 1             0
7008                        1                 1             0
7009                        0                 0             0
```

```
      prediction_label  prediction_score
7000                 0            0.9274
7001                 0            0.7149
7002                 0            0.9466
7003                 0            0.7444
7004                 0            0.7630
7005                 0            0.9452
7006                 0            0.7288
7007                 0            0.9550
7008                 1            0.5804
7009                 0            0.5804
```

0.10 Use python script to predict churn

[19]:
```
Code('predict_churn.py')
```

[19]:
```
import pandas as pd
from pycaret.classification import predict_model, load_model

def load_data(filepath):
    """
    Load churn data into a DataFrame from a given filepath.
    """
    return pd.read_csv(filepath)

def make_predictions(df, model_name='LDA'):
    """
    Use the specified PyCaret model to make predictions on the provided␣
 ↪DataFrame.
    """
    # Load the pre-trained PyCaret model
    model = load_model(model_name)

    # Make predictions on the DataFrame
    predictions = predict_model(model, data=df)

    # Rename and map the prediction labels
    predictions['Churn_prediction'] = predictions['prediction_label'].map({1:␣
 ↪'Churn', 0: 'No Churn'})

    return predictions['Churn_prediction']

if __name__ == "__main__":
    df = load_data('new_churn_data.csv')
    predictions = make_predictions(df, model_name='LDA')
    print('Predictions:')
    print(predictions)
```

[20]:
```
%run predict_churn.py
```

```
Transformation Pipeline and Model Successfully Loaded

<IPython.core.display.HTML object>

Predictions:
0    No Churn
1    No Churn
2    No Churn
3    No Churn
4    No Churn
5    No Churn
6    No Churn
7    No Churn
8       Churn
9    No Churn
Name: Churn_prediction, dtype: object
```

0.11 Summary

DS automation process begins with importation of necessary libraries and modules, encompassing pandas, pickle, and PyCaret functionalities tailored for classification tasks. Following this, the preprocessed churn data is imported from a CSV file into a pandas DataFrame. Subsequently, PyCaret's autoML environment is established, with the target variable specified as Churn.

Exploration of the autoML setup ensues by discerning its type and accessing specific elements. Various classification models undergo comparison, culminating in the selection of the best-performing model via PyCaret's compare_models function. Information pertaining to the optimal model is then retrieved, and a subset of the DataFrame (rows 7000-7010) is extracted for further analysis.

Utilizing the chosen model, predictions are generated for the selected rows, and subsequently, the model is saved under the name LDA. Utilizing the pickle serialization method, the model is stored and subsequently reloaded for subsequent analyses. A new dataset is created by duplicating the 10 rows while excluding the Churn column. Predictions are then made using the reloaded model, with a focus on computing the probability of churn for each new prediction.

Finally, the saved LDA model is loaded, and predictions are once again made for the new data. This comprehensive process illustrates a comprehensive workflow, encompassing data loading, setup, model selection, prediction generation, and supplementary analyses. The approach not only facilitates churn prediction but also provides deep insights into model performance.