Week5_Joythi_sanam

February 18, 2024

0.1 Import libraries and modules

[6]:
```
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

[7]:
```
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

[8]:
```
automl = setup(df, target='Churn')
```

```
<pandas.io.formats.style.Styler at 0x7fd9e503dc90>
```

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 4875, 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 4ad6.

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

[12]:
```
# Access the elements of the automl_setup object
automl_element = automl.get_config("X_train")
automl_element
```

[12]:

	tenure	PhoneService	Contract	MonthlyCharges	TotalCharges	\
4827	16	1	0	81.000000	1312.150024	
4167	50	1	1	75.699997	3876.199951	
249	42	1	1	99.000000	4298.450195	
145	65	1	3	99.050003	6416.700195	
1062	34	1	0	50.200001	1815.300049	
…	…	…	…	…	…	

```
2927       1              1         0       69.900002     69.900002
1240      25              1         0       20.150000    536.349976
3538      71              1         3      100.199997   7209.000000
4613      54              1         0       79.500000   4370.250000
3045      48              1         1       65.650002   3094.649902

      MonthlyCharges_to_tenure_Ratio  Bank transfer (automatic)  \
4827                        5.062500                          0
4167                        1.514000                          1
249                         2.357143                          0
145                         1.523846                          0
1062                        1.476471                          1
…                                …                          …
2927                       69.900002                          0
1240                        0.806000                          0
3538                        1.411268                          0
4613                        1.472222                          0
3045                        1.367708                          0

      Credit card (automatic)  Electronic check  Mailed check
4827                        0                 0             0
4167                        0                 1             0
249                         0                 0             0
145                         1                 1             0
1062                        0                 1             0
…                         …                 …             …
2927                        0                 0             0
1240                        0                 1             1
3538                        1                 1             0
4613                        0                 0             0
3045                        0                 0             0

[4930 rows x 10 columns]
```

0.4 compare classification models

[13]:
```
best_model = compare_models()
```

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

<pandas.io.formats.style.Styler at 0x7fd9e45c54d0>

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

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

The Linear Discriminant Analysis (LDA) model appears to have the highest accuracy among the models listed.

LDA (Linear Discriminant Analysis): Accuracy: 0.7955 AUC: 0.8346 Recall: 0.4587 Precision: 0.6662 F1 Score: 0.5427 Kappa: 0.4170 MCC: 0.4293 Training Time (Sec): 0.1760

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

[15]:
```
best_model
```

[15]:
```
LinearDiscriminantAnalysis(covariance_estimator=None, n_components=None,
                           priors=None, shrinkage=None, solver='svd',
                           store_covariance=False, tol=0.0001)
```

0.5 Select rows

[17]:
```
rows = df.iloc[500:510]
rows
```

[17]:

	tenure	PhoneService	Contract	MonthlyCharges	TotalCharges	Churn	\
500	34	1	1	116.25	3899.05	0	
501	71	1	3	80.70	5676.00	0	
502	70	1	1	65.20	4543.15	0	
503	52	1	0	84.05	4326.80	0	
504	69	1	3	79.45	5502.55	0	
505	20	1	0	94.10	1782.40	1	
506	11	1	0	78.00	851.80	0	
507	2	1	0	94.20	167.50	1	
508	6	1	0	80.50	502.85	1	
509	1	1	0	19.85	19.85	0	

	MonthlyCharges_to_tenure_Ratio	Bank transfer (automatic)	\
500	3.419118	0	
501	1.136620	0	
502	0.931429	1	
503	1.616346	1	
504	1.151449	0	
505	4.705000	0	
506	7.090909	0	
507	47.100000	0	
508	13.416667	0	
509	19.850000	0	

	Credit card (automatic)	Electronic check	Mailed check
500	1	1	0
501	1	1	0
502	0	1	0
503	0	1	0
504	1	1	0
505	0	0	0
506	0	1	1

507	0	0	0
508	0	0	0
509	0	1	1

0.6 Use best_model to predict churn for the rows

[18]:
```
predicted_rows = predict_model(best_model, rows)
predicted_rows
```

```
<pandas.io.formats.style.Styler at 0x7fd9e4548b90>
```

[18]:

	tenure	PhoneService	Contract	MonthlyCharges	TotalCharges	\
500	34	1	1	116.250000	3899.050049	
501	71	1	3	80.699997	5676.000000	
502	70	1	1	65.199997	4543.149902	
503	52	1	0	84.050003	4326.799805	
504	69	1	3	79.449997	5502.549805	
505	20	1	0	94.099998	1782.400024	
506	11	1	0	78.000000	851.799988	
507	2	1	0	94.199997	167.500000	
508	6	1	0	80.500000	502.850006	
509	1	1	0	19.850000	19.850000	

	MonthlyCharges_to_tenure_Ratio	Bank transfer (automatic)	\
500	3.419118	0	
501	1.136620	0	
502	0.931429	1	
503	1.616346	1	
504	1.151449	0	
505	4.705000	0	
506	7.090909	0	
507	47.099998	0	
508	13.416667	0	
509	19.850000	0	

	Credit card (automatic)	Electronic check	Mailed check	Churn	\
500	1	1	0	0	
501	1	1	0	0	
502	0	1	0	0	
503	0	1	0	0	
504	1	1	0	0	
505	0	0	0	1	
506	0	1	1	0	
507	0	0	0	1	
508	0	0	0	1	
509	0	1	1	0	

```
     prediction_label  prediction_score
500                 0            0.7025
501                 0            0.9704
502                 0            0.9493
503                 0            0.8606
504                 0            0.9696
505                 1            0.5729
506                 0            0.6743
507                 1            0.9179
508                 1            0.6583
509                 0            0.8620
```

We predict the selected rows using the Linear Discriminant Analysis (LDA) model.

LDA Model Performance Metrics:

```
Accuracy: 1.0000
AUC: 1.0000
Recall: 1.0000
Precision: 1.0000
F1 Score: 1.0000
Kappa: 1.0000
MCC: 1.0000
```

These perfect 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.

Notably, the accuracy of 1.0000 suggests that the model correctly predicted the target variable for each of the rows.

0.7 save model and serialize

[19]:
```
save_model(best_model, 'LDA')
```

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

[19]:
```
(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')
```

[21]:
```
with open('LDA_model.pk', 'wb') as f:
    pickle.dump(best_model, f)
```

[22]:
```
with open('LDA_model.pk', 'rb') as f:
    loaded_model = pickle.load(f)
```

0.8 Create new data and save to csv

[23]:
```
new_data = rows.copy()
new_data.drop('Churn', axis=1, inplace=True)
new_data.to_csv('new_churn_data.csv', index=False)
```

[24]:
```
loaded_model_prediction = loaded_model.predict(new_data)
loaded_model_prediction
```

[24]:
```
array([0, 0, 0, 0, 0, 1, 0, 1, 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

[26]:
```
probability_of_churn = loaded_model.predict_proba(new_data)[:, 1]
probability_of_churn
```

[26]:
```
array([0.29749369, 0.02959457, 0.05072708, 0.13935076, 0.03040274,
       0.57289406, 0.32565703, 0.91791234, 0.65828767, 0.13795148])
```

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 0.5729 suggests a 57.29% likelihood of churn for the sixth row in new_data.

0.10 Get percentile

[35]:
```
df_no_missing = df.dropna()

percentile_rank = (
    (loaded_model.predict_proba(df_no_missing.drop('Churn', axis=1))[:, 1] <=␣
 ↪probability_of_churn).mean() * 100
)
percentile_rank
```

[35]:
```
100.0
```

The percentile_rank is showing 100.0. This suggests that the predicted probability of churn for the new data point is at the highest end of the distribution of probability predictions from the training dataset. A percentile rank of 100.0 indicates that the predicted probability is equal to or greater than all the probabilities in the training dataset.

In this context, where the target variable is binary (churn or not churn), a high predicted probability of churn (close to 1.0) often suggests a high level of confidence by the model that the new data point belongs to the positive class (churn).

[38]:
```
loaded_lda = load_model('LDA')
```

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

[40]:
```
loaded_lda_prediction = predict_model(loaded_lda, new_data)
loaded_lda_prediction
```

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

[40]:

	tenure	PhoneService	Contract	MonthlyCharges	TotalCharges	\
500	34	1	1	116.250000	3899.050049	
501	71	1	3	80.699997	5676.000000	
502	70	1	1	65.199997	4543.149902	
503	52	1	0	84.050003	4326.799805	
504	69	1	3	79.449997	5502.549805	
505	20	1	0	94.099998	1782.400024	
506	11	1	0	78.000000	851.799988	
507	2	1	0	94.199997	167.500000	
508	6	1	0	80.500000	502.850006	
509	1	1	0	19.850000	19.850000	

	MonthlyCharges_to_tenure_Ratio	Bank transfer (automatic)	\
500	3.419118	0	
501	1.136620	0	
502	0.931429	1	
503	1.616346	1	
504	1.151449	0	
505	4.705000	0	
506	7.090909	0	

```
507                       47.099998                          0
508                       13.416667                          0
509                       19.850000                          0

     Credit card (automatic)  Electronic check  Mailed check  \
500                        1                 1             0
501                        1                 1             0
502                        0                 1             0
503                        0                 1             0
504                        1                 1             0
505                        0                 0             0
506                        0                 1             1
507                        0                 0             0
508                        0                 0             0
509                        0                 1             1

     prediction_label  prediction_score
500                 0            0.7025
501                 0            0.9704
502                 0            0.9493
503                 0            0.8606
504                 0            0.9696
505                 1            0.5729
506                 0            0.6743
507                 1            0.9179
508                 1            0.6583
509                 0            0.8620
```

0.11 Load external python script to predict churn

[41]:
```
Code('predict_churn.py')
```

[41]:
```
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)
```

[42]:
```
%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       Churn
6    No Churn
7       Churn
8       Churn
9    No Churn
Name: Churn_prediction, dtype: object
```

The `No Churn` and `Churn` labels in the `Churn_prediction` column indicate whether each corresponding entry is predicted as a churn or non-churn instance.

0.12 Summary

The DS automation process begins by importing essential libraries and modules, such as pandas, pickle, and PyCaret functions for classification purposes. The cleaned churn data is loaded from a CSV file into a pandas DataFrame, and PyCaret's autoML environment is set up with the target variable specified as Churn. The autoML setup is explored by determining its type and accessing specific elements. A variety of classification models are compared, and the best-performing model is selected through PyCaret's compare_models function. Information about the best-performing model is then retrieved, and 10 rows (500-509) from the DataFrame are extracted.

Using the best model, predictions are made for the selected row, and the model is saved with the name LDA. The serialized model is stored using pickle, and later, it is loaded back for further

analysis. A new dataset is created by copying the 10 rows and excluding the Churn column. Predictions are made with the loaded model, including returning the probability of churn for each new prediction. Additional analysis involves calculating the percentile rank of the prediction within the distribution of probability predictions from the training dataset.

Finally, we load the saved LDA model and predictions are made for the new data. This is a demonstration of the end-to-end workflow, from loading and setting up data to model selection, prediction, and additional analyses, providing a robust approach to churn prediction with detailed insights into model performance.