In [14]:
```
import pandas as pd
```

In [2]:
```
from pycaret.classification import setup, compare_models, evaluate_model, save_model, load_model, predict_model
```

In [3]:
```
churn_data = pd.read_csv('/content/updated_churn_data (1).csv')
```

In [4]:
```
clf = setup(data=churn_data, target='Churn')
```

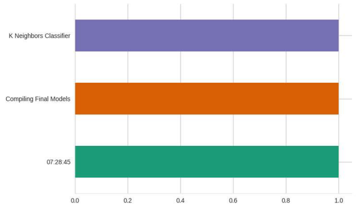
	Description	Value
0	Session id	2877
1	Target	Churn
2	Target type	Binary
3	Original data shape	(7043, 8)
4	Transformed data shape	(7043, 13)
5	Transformed train set shape	(4930, 13)
6	Transformed test set shape	(2113, 13)
7	Numeric features	4
8	Categorical features	3
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	c9e7

In [5]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
knn	K Neighbors Classifier	0.7556	0.7110	0.3822	0.5589	0.4526	0.3027	0.3124	0.1560
dt	Decision Tree Classifier	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1230
ridge	Ridge Classifier	0.7347	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1800
rf	Random Forest Classifier	0.7347	0.6284	0.0000	0.0000	0.0000	0.0000	0.0000	0.4420
qda	Quadratic Discriminant Analysis	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1740
ada	Ada Boost Classifier	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1260
gbc	Gradient Boosting Classifier	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4540
lda	Linear Discriminant Analysis	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1250
et	Extra Trees Classifier	0.7347	0.5099	0.0000	0.0000	0.0000	0.0000	0.0000	0.3200
xgboost	Extreme Gradient Boosting	0.7347	0.6838	0.0000	0.0000	0.0000	0.0000	0.0000	0.1620
lightgbm	Light Gradient Boosting Machine	0.7347	0.4830	0.0000	0.0000	0.0000	0.0000	0.0000	0.4600
dummy	Dummy Classifier	0.7347	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1350
lr	Logistic Regression	0.7341	0.8107	0.0000	0.0000	0.0000	-0.0012	-0.0047	1.0500
nb	Naive Bayes	0.6629	0.8057	0.8693	0.4333	0.5780	0.3465	0.4054	0.1250
svm	SVM - Linear Kernel	0.5057	0.0000	0.6557	0.2124	0.3203	0.0648	0.1199	0.1860

```
Processing:   0%|          | 0/65 [00:00<?, ?it/s]
```

Categorical distributions



In [6]:
```
evaluate_model(best_model)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',), options=(('Pipeline Plot', 'pipelin…
```

In [7]:
```
save_model(best_model, 'best_churn_model')
```

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

Out[7]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['tenure', 'PhoneService',
                                              'MonthlyCharges', 'TotalCharges'],
                                     transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_value=None,
                                                               keep_empty_features=False,
                                                               missing_values=nan,
                                                               strategy='mean',
                                                               verbose='deprecated'))),
                 ('categorical_imputer',
                  Transf...
                                     transformer=TargetEncoder(cols=['customerID'],
                                                               drop_invariant=False,
                                                               handle_missing='return_nan',
                                                               handle_unknown='value',
                                                               hierarchy=None,
                                                               min_samples_leaf=20,
                                                               return_df=True,
                                                               smoothing=10,
                                                               verbose=0))),
                 ('trained_model',
                  KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                       metric='minkowski', metric_params=None,
                                       n_jobs=-1, n_neighbors=5, p=2,
                                       weights='uniform'))],
          verbose=False),
 'best_churn_model.pkl')
```

In [11]:
```
def predict_churn_probability_whole_dataset(data):
  model = load_model('best_churn_model')

  predictions = predict_model(model, data=churn_data)

  return predictions
```

In [9]:
```
new_data=pd.read_csv('/content/new_churn_data.csv')
```

In [12]:
```
predictions = predict_churn_probability_whole_dataset(new_data)
```

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

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	K Neighbors Classifier	0.8194	0.8536	0.5243	0.7190	0.6064	0.4929	0.5033

In [13]:
```
print(predictions)
```

```
     customerID  tenure  PhoneService        Contract  \
0    7590-VHVEG       1             0  Month-to-month
1    5575-GNVDE      34             1        One year
2    3668-QPYBK       2             1  Month-to-month
3    7795-CFOCW      45             0        One year
4    9237-HQITU       2             1  Month-to-month
...         ...     ...           ...             ...
7038 6840-RESVB      24             1        One year
7039 2234-XADUH      72             1        One year
7040 4801-JZAZL      11             0  Month-to-month
7041 8361-LTMKD       4             1  Month-to-month
7042 3186-AJIEK      66             1        Two year

                 PaymentMethod  MonthlyCharges  TotalCharges  Churn  \
0             Electronic check       49.650002     49.650002      0
1                 Mailed check       55.573528   1889.500000      0
2                 Mailed check       54.075001    108.150002      1
3    Bank transfer (automatic)       40.905556   1840.750000      0
4             Electronic check       75.824997    151.649994      1
...                        ...             ...           ...    ...
7038              Mailed check       82.937500   1990.500000      0
7039   Credit card (automatic)       96.878471   6975.250000      0
7040          Electronic check       31.495455    346.450012      0
7041              Mailed check       76.650002    306.600006      1
7042 Bank transfer (automatic)      103.704544   6844.500000      0

     prediction_label  prediction_score
0                   0               0.8
1                   0               1.0
2                   0               0.8
3                   0               1.0
4                   1               0.6
...               ...               ...
7038                0               0.8
7039                0               1.0
7040                0               0.8
7041                1               0.8
7042                0               0.8

[7043 rows x 10 columns]
```

GITHUB LINK

https://github.com/nishadas1/Data-Science- (https://github.com/nishadas1/Data-Science-)

Summary

In this week's task, we utilized the PyCaret library to develop a predictive model for customer churn based on a dataset containing various customer attributes such as tenure, phone service, contract details, and payment method. After preprocessing the data and setting up the PyCaret environment, we compared several machine learning algorithms to identify the best-performing model. Upon selecting the optimal model, we evaluated its performance metrics, saved it for future use, and created a function to predict churn probabilities for new data. This streamlined approach facilitated efficient model development and evaluation, enabling businesses to make data-driven decisions to mitigate customer churn and improve customer retention strategies.