In [20]:
```
import pandas as pd

df = pd.read_csv('prepped_diabetes_data.csv', index_col='Patient number')
```

Out[20]:

Patient number	Cholesterol	Glucose	HDL Chol	Age	Gender	Height	Weight	BMI	Systolic BP	Diastolic BP	wai
1	193	77	49	19	1	61	119	22.5	118	70	
2	146	79	41	19	1	60	135	26.4	108	58	
3	217	75	54	20	1	67	187	29.3	110	72	
4	226	97	70	20	1	64	114	19.6	122	64	
5	164	91	67	20	1	70	141	20.2	122	86	
...	...	...	...	...	...	...	...	...	...	...	
386	227	105	44	83	1	59	125	25.2	150	90	
387	226	279	52	84	1	60	192	37.5	144	88	
388	301	90	118	89	1	61	115	21.7	218	90	
389	232	184	114	91	1	61	127	24.0	170	82	
390	165	94	69	92	1	62	217	39.7	160	82	

390 rows × 15 columns

In [21]:

In [24]:

	Description	Value
0	session_id	362
1	Target	Diabetes
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(390, 15)
5	Missing Values	False
6	Numeric Features	13
7	Categorical Features	1
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(272, 14)

In [30]:

Out[30]:

Patient number	Cholesterol	Glucose	HDL Chol	Age	Height	Weight	BMI	Systolic BP	Diastolic BP	waist
8	164.0	71.0	63.0	20.0	72.0	145.0	19.700001	108.0	78.0	29.0
169	269.0	59.0	66.0	41.0	67.0	191.0	29.900000	130.0	73.0	38.0
51	164.0	94.0	58.0	28.0	67.0	180.0	28.200001	128.0	94.0	39.0
242	218.0	182.0	54.0	51.0	66.0	215.0	34.700001	139.0	69.0	42.0
260	216.0	79.0	46.0	54.0	65.0	138.0	23.000000	132.0	80.0	33.0
...	...	...	...	...	...	...	...	...	...	...
95	300.0	65.0	59.0	34.0	65.0	160.0	26.600000	120.0	60.0	40.0
137	268.0	90.0	48.0	38.0	63.0	181.0	32.099998	142.0	100.0	38.0
112	179.0	81.0	35.0	36.0	63.0	125.0	22.100000	110.0	76.0	33.0
191	244.0	101.0	39.0	44.0	71.0	168.0	23.400000	140.0	89.0	36.0
30	164.0	86.0	40.0	23.0	69.0	245.0	36.200001	126.0	75.0	44.0

118 rows × 14 columns

In [31]:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.9081	0.9132	0.5700	0.8633	0.6578	0.6111	0.6427	0.0150
knn	K Neighbors Classifier	0.9007	0.8880	0.5650	0.8067	0.6428	0.5893	0.6140	0.0340
ridge	Ridge Classifier	0.9007	0.0000	0.5050	0.7767	0.5993	0.5562	0.5805	0.0150
catboost	CatBoost Classifier	0.8937	0.9295	0.6150	0.7350	0.6548	0.5940	0.6056	1.4950
et	Extra Trees Classifier	0.8898	0.8908	0.4850	0.8133	0.5776	0.5238	0.5590	0.1300
qda	Quadratic Discriminant Analysis	0.8827	0.7754	0.5300	0.6300	0.5685	0.5120	0.5189	0.0210
lightgbm	Light Gradient Boosting Machine	0.8827	0.9178	0.5550	0.7133	0.5962	0.5337	0.5517	0.1560
xgboost	Extreme Gradient Boosting	0.8753	0.9082	0.5550	0.6855	0.5933	0.5235	0.5385	0.1870
rf	Random Forest Classifier	0.8751	0.9051	0.4900	0.7533	0.5593	0.4946	0.5246	0.1570
gbc	Gradient Boosting Classifier	0.8643	0.8919	0.5500	0.6562	0.5822	0.5034	0.5154	0.0590
nb	Naive Bayes	0.8639	0.8906	0.6100	0.6690	0.5870	0.5118	0.5401	0.0160

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.8565	0.8254	0.5350	0.5400	0.5322	0.4578	0.4611	0.0780
dt	Decision Tree Classifier	0.8349	0.6961	0.4850	0.5458	0.5004	0.4051	0.4128	0.0180

In [38]:

Out[38]: sklearn.discriminant_analysis.LinearDiscriminantAnalysis

In [36]:

Out[36]: (1, 15)

In [37]:

Out[37]:

Patient number	Cholesterol	Glucose	HDL Chol	Age	Gender	Height	Weight	BMI	Systolic BP	Diastolic BP	wai
389	232	184	114	91	1	61	127	24.0	170	82	

In [39]:

```
Transformation Pipeline and Model Succesfully Saved
```

Out[39]:

In [11]:
```
import pickle

with open('LDA_model.pk', 'wb') as f:
```

In [12]:
```
with open('LDA_model.pk', 'rb') as f:
```

In [13]:
```
new_data = df.iloc[-2:-1].copy()
new_data.drop('Diabetes', axis=1, inplace=True)
```

Out[13]:
```
array([1])
```

In [41]:

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

In [42]:

Out[42]:

Patient number	Cholesterol	Glucose	HDL Chol	Age	Gender	Height	Weight	BMI	Systolic BP	Diastolic BP	wa
389	232	184	114	91	1	61	127	24.0	170	82	

In [48]:
```
from IPython.display import Code
```

Out[48]:

```
import pandas as pd
from pycaret.classification import predict_model, load_model

def load_data(filepath):
    """
    Loads diabetes data into a DataFrame from a string filepath.
    """
    df = pd.read_csv(filepath, index_col='Patient number')
    return df
```

In [49]:

```
Transformation Pipeline and Model Successfully Loaded
predictions:
Patient number
391       Diabetes
392    No diabetes
393    No diabetes
394    No diabetes
Name: Diabetes_prediction, dtype: object
```

Summary and Short Analysis

This week assignment is about automation techniques for data science with Python. First we loaded the same prepared data from week 2 where everything ha been converted to numbers. Second, we used pycart for autoML but before doing that we installed Python package with condo. Then we import the function we need. After that, we satup our autoML and we basically run the autoML to find the best model.But before we run our autoML to find the best model, we have if our datatypes of the input are correct and in our case, they were fine. This i updated in real time within the notebook as needed.
Boosting algorithms such as xgboost and catboost take the longest to run. xgboost is frequently towards the top. To use xgboost and lightgbm, we must either enable preprocessing (which transforms categorical columns to numeric columns) or set our categorical columns to numeric using 'automl = setup(df, target='Diabetes', preprocess=False, numeric features= Gender). Our best model object now contains the highest-scoring model.We can also use the 'compare models' argument'sort' to select another metric as our scoring metric.
It employs precision by default (and we can see the table above is sorted by accuracy). We may set this to'sort='Precision" to utilize precision (TP / (TP FN)), for example. we found out that our best mpdel is LDA. Third, we used our best_model to make predictions. We selected the last row, but using the indexing [-2:-1] to make it 2D array instead of 1D. From there, we can see thi creates a new column, 'Score' with the propability of class1. It also creates 'Label' column with the predicted label, where it rounds up if score is >=0.5 (greater than or equal to 0.5). Finally,we stored our trained model so we can utilize it later in a Python file.For instance, pycaret includes a convenient method that saves the model as a pickle file. In this case, we utilize the built-in 'open' function to open a file called 'LDA model.pk,' then open it fo writing with "w" and in binary format with "b."That file object is saved in th variable 'f.'After we exit the with statement, the file is automatically closed; otherwise, we must use the method 'close' from the file object 'f'. Pickle is then used to store our data to a file. After that, we saved our code to GitHub.