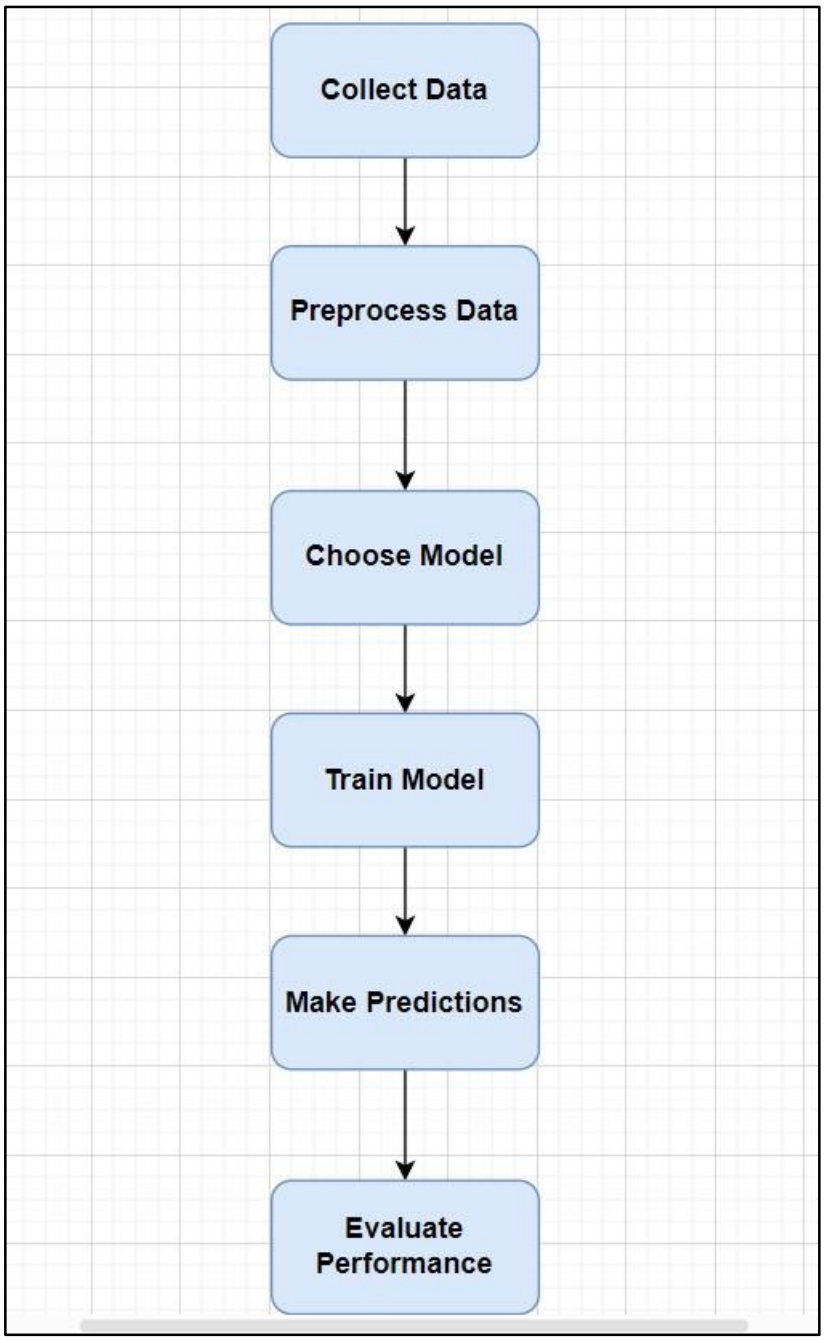

Collect Data
Preprocess Data
Choose Model
Train Model
Make Predictions
Evaluate Performance

Import Required Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from imblearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from imblearn.over_sampling import SMOTE
from sklearn import set_config
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# Set display options to show the full pipeline
set_config(display='diagram')
```

Load dataset

```
diabetes_df = pd.read_csv('diabetes_prediction_dataset.csv')
diabetes_df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

```
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

Check for null values

```
diabetes_df.isnull().sum()
```

```
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

```
diabetes_df.diabetes.value_counts()
```

```
0    91500
1     8500
Name: diabetes, dtype: int64
```

Our dataset is imbalance will handle that later 😎

```
numeric_features = ['age','hypertension', 'heart_disease','bmi', 'HbA1c_level','blood_glucose_level']
categorical_features = ['gender','smoking_history']
```

```
X = diabetes_df.drop('diabetes',axis=1)
y = diabetes_df['diabetes']
```

```
# X contains feature data and y contains target labels
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)

print(f'Size of training data: {X_train.shape}')
print(f'Size of testing  data: {X_test.shape}')
```

```
Size of training data: (80000, 8)
Size of testing  data: (20000, 8)
```

Create pipelines for numerical and categorical features

```
# Define separate pipelines for numeric and categorical features
#StandardScaler for numeric features
numeric_transformer = Pipeline([
    ('scaler', StandardScaler())
])

# OneHotEncoder for categorical features
# Specify which columns are numeric and categorical
categorical_transformer = Pipeline([
    ('encoder', OneHotEncoder())
])

# Define a ColumnTransformer to combine the pipelines (same as before)
preprocessor = ColumnTransformer([
    ('numeric', numeric_transformer, numeric_features),
    ('categorical', categorical_transformer, categorical_features)
])
```

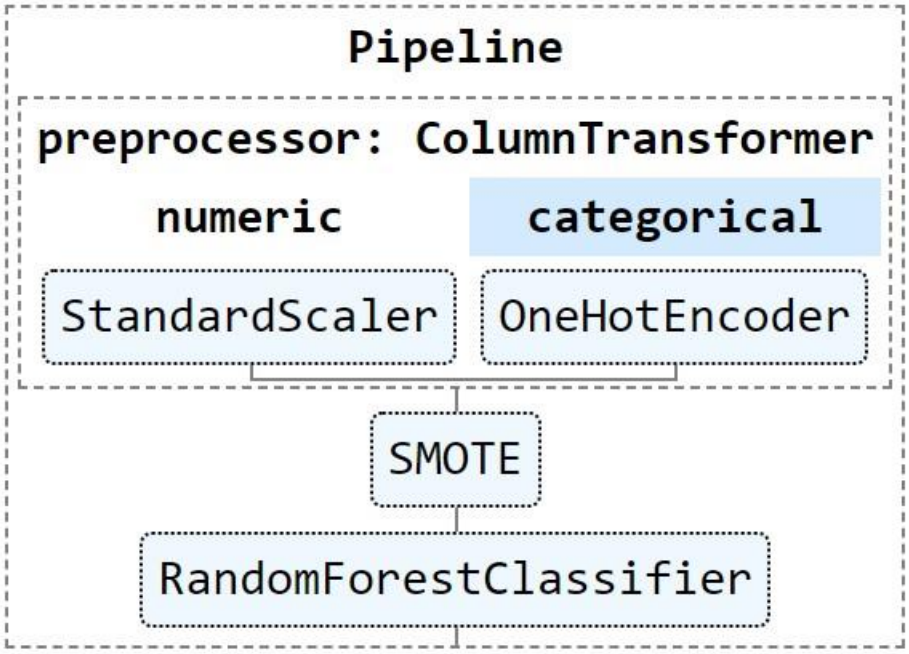
```
# Create an SMOTE instance
#SMOTE can help alleviate class imbalance issues
smt = SMOTE(random_state=42)
```

Add a model to a final pipeline

```
# Create a final pipeline with the preprocessor, SMOTE, and your chosen model(Random Forest)
final_pipeline = Pipeline([
    ('preprocessor',preprocessor),
    ('smote',smt),
    ('classifier',RandomForestClassifier())
])
```

Display the pipeline

```
final_pipeline
```



Pass data through the pipeline

```
# Fit the final pipeline on the training data
final_pipeline.fit(X_train, y_train)

# Evaluate the final pipeline on the test data (unseen data)
accuracy = final_pipeline.score(X_test, y_test)
y_pred = final_pipeline.predict(X_test)
print("Test Accuracy:", accuracy)
```

```
Test Accuracy: 0.95785
```

```
class_report = classification_report(y_test, y_pred)
print("Classification Report:\n", class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     18292
           1       0.75      0.75      0.75      1708

    accuracy                           0.96     20000
   macro avg       0.87      0.86      0.86     20000
weighted avg       0.96      0.96      0.96     20000
```

Plotting a confusion matrix

```
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```