What is pipeline?

A pipeline in machine learning is way to streamline a lot of routing processes, making it easier to produce a model that can handle different inputs and different cases. The concept of a pipeline is to define a series of data processing steps, where each step is a tuple containing a name (a string) and an instance of an estimator or a transformer. The pipeline then chains these steps together so that the output of one step becomes the input to the next.

Chaining steps:

* A pipeline allows you to define a series of processing steps, including data preprocessing, feature engineering and model training in sequential manner.

Consistency across train and test:

* Pipelines make it easy to ensure consistency in data processing between training and test sets. All the transformation applied to training data can be replicated to test data using the same pipelines.

Ease of Deployment:

* When you deploy a machine learning model, especially in a production environment, you often need to apply the same preprocessing steps in incoming data before making predictions. A pipeline makes this process seamless and ensure that the data is pre-processed in the same way as during training.

Readability and Maintainability:

* Pipelines enhance the readability and maintainability of your code. Instead of applying preprocessing steps separately on training and test data, you can encapsulate all the steps in a single pipeline, making the code more modular and easier to understand.

Code:

From sklearn.pipeline import Pipeline, make\_pipeline

From sklearn.preprocessing import StandardScaler, OneHotEncoder

From sklearn.impute import SimpleImputer

From sklearn.feature\_selection import SelectKBest, chi2

From sklearn.compose import ColumnTransformer

From sklearn.decomposition import PCA

From sklearn.ensemble import RandomForestClassifier

# Imputations transformer for missing values

Trf1 = ColumnTransformer([

(‘impute\_col1’,SimpleImputer(),[2]), # No 2 is the 2nd column index

(‘impute\_col2’,SimpleImputer(),[6]) # No 6 is the 6th column index

], remainder=’passthrough’) # remainder is set as passthrough else rest all the columns will be dropped

# One hot encoder

Trf2 = ColumnTransformer([

(‘ohe\_col1’,OneHotEncoder(sparse=False,handle\_unknown=’ignore’),[1,6]),

Remainder=’passthrough’

])

Trf3 = ColumnTransformer([

(‘scale’,StandardScaler(),slice(0,10)) # Slice will select all the columns

])

Trf4 = RandomForestClassifier()

Pipe = Pipeline([

(‘trf1’, trf1),

(‘trf2’, trf2),

(‘trf3’, trf3),

(‘trf4’,trf40)

])

Pipeline = Pipeline([

(‘scaler’, StandardScaler()),

(‘pca’,PCA(n\_components=5)),

(‘classifier’,’RandomForestClassifier’())

])

Pipeline.fit(X\_train,y\_train)

If we are just updating pre-processing steps and not the model in the pipeline then we have to call function pipeline.transform instead.

Predictions = pipeline.predict(X\_test)

To visualize the Pipeline, we have to pass the below code:

We can simply export the pipeline with pickle

Import pickle

Pickle.dump(pipe,open(‘pipe.pkl’,’wb’)

Pickle.load(open(‘pipe.pkl’,’rb’)

Pipe.predict(test\_input)