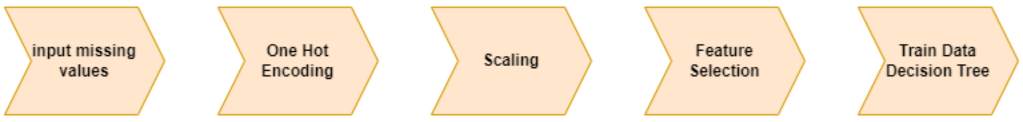
USING PIPELINE

Firstly I will be importing the neccesary libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

Plan



We have 2 missing columns **'Age'** and **'Embarked'**

```
titanic_dataset = pd.read_csv('titanic_dataset.csv')
```

```
titanic_dataset.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
titanic_dataset.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic_dataset
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Let's drop down columns 'PassengerID', 'Name', 'Ticket', 'Cabin' as these don't signify or play any role in predicting Passenger's Survival.

```
titanic_dataset.drop(columns=['PassengerId','Name','Ticket','Cabin'],inplace=True)
```

```
titanic_dataset
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	NaN	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

891 rows × 8 columns

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(titanic_dataset.drop(columns=['Survived']),
                                                    titanic_dataset['Survived'],
                                                    test_size=0.2,
                                                   random_state=42)
```

Train Test Split the data considering target variable as 'Survive'

1. Impute Transformer

```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline,make_pipeline
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.tree import DecisionTreeClassifier
```

```
#_________________________________________________________Imputation Transformer______________________________________________________________

trf1 = ColumnTransformer ([('impute_age', SimpleImputer(), [2]),
                           ('impute_embarked', SimpleImputer(strategy='most_frequent'), [6])],
                          remainder='passthrough')
```

Here I passed list of tuples,
(i) **impute_age** --> Simple Imputer object and its aim is to fill null values with mean which I applied on column [2] i.e., 'Age'.
(ii) **impute_embarked** --> SimpleImputer object and its aim is to fill null values with most frequent values which I applied on column [6] i.e., 'Embarked'.
This technique automatically fills null values using Simple Imputer.

NOTE: Here I didn't use column name and instead I use column number because after imputation it doesn't be in form of dataframe, it is in format of numpy array.

NOTE: Use column number instead of column name while using Pipeline.
And for other columns we did passthrough, else the other columns would has defaultly be removed.

```
X_train
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
331	1	male	45.5	0	0	28.5000	S
733	2	male	23.0	0	0	13.0000	S
382	3	male	32.0	0	0	7.9250	S
704	3	male	26.0	1	0	7.8542	S
813	3	female	6.0	4	2	31.2750	S
...	...	...	...	...	...	...	...
106	3	female	21.0	0	0	7.6500	S
270	1	male	NaN	0	0	31.0000	S
860	3	male	41.0	2	0	14.1083	S
435	1	female	14.0	1	2	120.0000	S
102	1	male	21.0	0	1	77.2875	S

712 rows × 7 columns

```
X_test
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
709	3	male	NaN	1	1	15.2458	C
439	2	male	31.0	0	0	10.5000	S
840	3	male	20.0	0	0	7.9250	S
720	2	female	6.0	0	1	33.0000	S
39	3	female	14.0	1	0	11.2417	C
...	...	...	...	...	...	...	...
433	3	male	17.0	0	0	7.1250	S
773	3	male	NaN	0	0	7.2250	C
25	3	female	38.0	1	5	31.3875	S
84	2	female	17.0	0	0	10.5000	S
10	3	female	4.0	1	1	16.7000	S

179 rows × 7 columns

```
y_train
```

```
331    0
733    0
382    0
704    0
813    0
      ..
106    1
270    0
860    0
435    1
102    0
Name: Survived, Length: 712, dtype: int64
```

```
y_test
```

```
709    1
439    0
840    0
720    1
39     1
      ..
433    0
773    0
25     1
84     1
10     1
Name: Survived, Length: 179, dtype: int64
```

2. One Hot Encoding

```
#____________________________________________________________One Hot Encoding______________________________________________________________
trf2 = ColumnTransformer([('OHE_sex_Embarked', OneHotEncoder(sparse=False, handle_unknown='ignore'), [1, 6])],
                         remainder='passthrough')
```

Here I created tuple OHE_sex_Embarked --> OneHotEncoder object, we can apply this technique on both columns 'Sex' and 'Embarked' as everything is inside data and not creating any new numpy array.
I applied One Hot Encoding on column 1 and 6, i.e 'Sex and 'Embarked'.

3. Scaling

```
#__________________________________________________________________Scaling__________________________________________________________________

trf3 = ColumnTransformer([('scale', MinMaxScaler(), slice(0, 8))])
```

Created tuple scale --> MinMaScaler object
Here I used MinMaxScaler because further I am going to do feature selection, else I had used Standard Scaler.
slice --> used to apply the scaling on all columns.
Because after One Hot Encoding 2 columns 'Sex' and 'Embarked' will be removed and their unique values/categories columns will be generated.
Sex --> Male, Female i.e., 2 categories i.e., 2 new columns
Embarked --> S, C, Q i.e., 3 categories i.e., 3 new columns. Therefore new total 5 new columns will get generated.

So code must be like
trf3 = ColumnTransformer([('scale', MinMaxScaler(), slice(0, 10))])

```
trf3 = ColumnTransformer([('scale', MinMaxScaler(), slice(0, 10))])
```

4. Feature Selection

```
#____________________________________________________________Feature Selection_____________________________________________________________

trf4= SelectKBest(score_func=chi2, k=8)
```

k = 8 means it will scale top 8 important features.

Decision Tree Classification

```
trf5 = DecisionTreeClassifier()
```

Now, as I created individual, now I will connect/assemble it using Pipeline

```
Pipe = Pipeline([('trf1', trf1), ('trf2', trf2), ('trf3', trf3), ('trf4', trf4), ('trf5', trf5)])
```

I just use Pipelinr model and created list of tuples where I enter name of that particular transformation and it's object.

```
from sklearn import set_config
```

```
set_config(display='diagram')
```

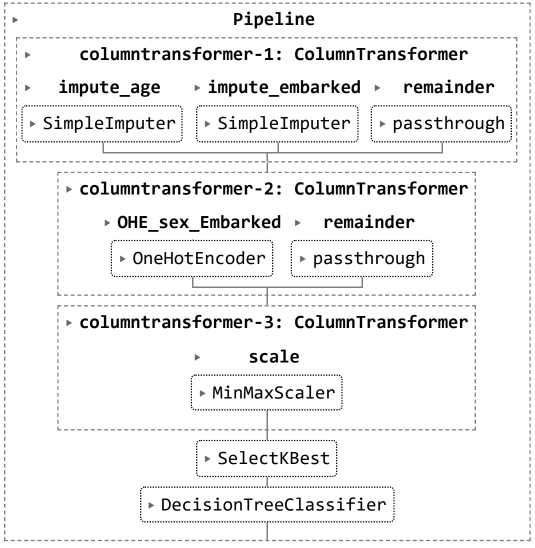
Pipeline VS make_pipeline

```
# make_pipeline

Pipe_make = make_pipeline(trf1, trf2, trf3, trf4, trf5)
```

```
Pipe_make.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
```



Here I just fit the model inside pipe that is Decision Tree on X_train and y_train.

If no algorithm was applied into our pipeline then I would had did as:
Pipe_make.fit_transform

For getting vizualization of Pipeline after getting trained we use:
set_config(display='diagram')

```
from sklearn import set_config
```

```
set_config(display='diagram')
```

```
Pipe_make.named_steps
```

```
{'columntransformer-1': ColumnTransformer(remainder='passthrough',
                   transformers=[('impute_age', SimpleImputer(), [2]),
                                 ('impute_embarked',
                                  SimpleImputer(strategy='most_frequent'),
                                  [6])]),
 'columntransformer-2': ColumnTransformer(remainder='passthrough',
                   transformers=[('OHE_sex_Embarked',
                                  OneHotEncoder(handle_unknown='ignore',
                                                sparse=False),
                                  [1, 6])]),
 'columntransformer-3': ColumnTransformer(transformers=[('scale', MinMaxScaler(), slice(0, 10, None))]),
 'selectkbest': SelectKBest(k=8, score_func=<function chi2 at 0x7d6bfd6c49d0>),
 'decisiontreeclassifier': DecisionTreeClassifier()}
```

Pipe_make.named_steps
It gives all the steps included in our Pipeline

So, for these reason we can use Pipeline, because it gives information about each tuple and performance inside it.
Foe e.g., It shows about our created tuples 'trf1', 'trf2', 'trf3', 'trf4', 'trf5'.

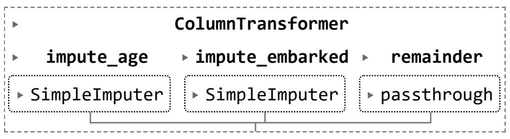
If we want to see what is happening inside 'trf1'
Pipe_make.named_steps['trf1']
So let's just implement it...

Let's see the use of Pipeline over make_pipeline

```
Pipe.named_steps
```

```
{'trf1': ColumnTransformer(remainder='passthrough',
                   transformers=[('impute_age', SimpleImputer(), [2]),
                                 ('impute_embarked',
                                  SimpleImputer(strategy='most_frequent'),
                                  [6])]),
 'trf2': ColumnTransformer(remainder='passthrough',
                   transformers=[('OHE_sex_Embarked',
                                  OneHotEncoder(handle_unknown='ignore',
                                                sparse=False),
                                  [1, 6])]),
 'trf3': ColumnTransformer(transformers=[('scale', MinMaxScaler(), slice(0, 10, None))]),
 'trf4': SelectKBest(k=8, score_func=<function chi2 at 0x7d6bfd6c49d0>),
 'trf5': DecisionTreeClassifier()}
```

```
Pipe.named_steps['trf1']
```



```
Pipe.named_steps['trf1'].transformers_
```

```
[('impute_age', SimpleImputer(), [2]),
 ('impute_embarked', SimpleImputer(strategy='most_frequent'), [6]),
 ('remainder', 'passthrough', [0, 1, 3, 4, 5])]
```

So, it gives the transformers used in trf1

If I want that what was the mean value for impute_age as which we implemented earlier.

```
Pipe.named_steps['trf1'].transformers_[0]
```

```
('impute_age', SimpleImputer(), [2])
```

```
Pipe.named_steps['trf1'].transformers_[0][1]
```

```
▾ SimpleImputer
SimpleImputer()
```

It shows which imputer was used
We can observe from the diagram, Like 0th column 1st step of flowchart (1st row).

```
Pipe.named_steps['trf1'].transformers_[1]
```

```
('impute_embarked', SimpleImputer(strategy='most_frequent'), [6])
```

```
Pipe.named_steps['trf1'].transformers_[1][1]
```

```
▾             SimpleImputer
SimpleImputer(strategy='most_frequent')
```

```
Pipe.named_steps['trf1'].transformers_[2]
```

```
('remainder', 'passthrough', [0, 1, 3, 4, 5])
```

```
Pipe.named_steps['trf1'].transformers_[2][1]
```

```
'passthrough'
```

```
Pipe.named_steps['trf1'].transformers_[0][1].statistics_
```

```
array([29.49884615])
```

So it gives the mean value.

Now, let's predict...

```
y_pred_ = Pipe.predict(X_test)
```

Let's check accuracy,

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, y_pred_)
```

```
0.6256983240223464
```

So here we can observe the accuracy coming out is around 62.5%

Let us Cross validate it:
Cross validation => different times we cross the values and train test split it, run the algorithm then calculate mean score for the accuracy

```
from sklearn.model_selection import cross_val_score
```

```
cross_val_score(Pipe, X_train, y_train, cv=5, scoring='accuracy')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
array([0.6013986 , 0.62237762, 0.68309859, 0.65492958, 0.63380282])
```

Here I calculated result by crossing the values 5 times and got the result as 60.1%, 62.2%, 68.3%, 65.4%, 63.3%.

```
cross_val_score(Pipe, X_train, y_train, cv=5, scoring='accuracy').mean()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
0.6391214419383433
```

Then just calcuated the mean of it so that we can get an average accuracy of our model's performance which comes out to be 63.9%.

HYPERPARAMETER TUNNING

GridSearch

```
# gridsearchcv
```

```
params = {'trf5__max_depth': [1, 2, 3, 4, 5, None]}
```

Here I declared values for max depth after creating parameters.

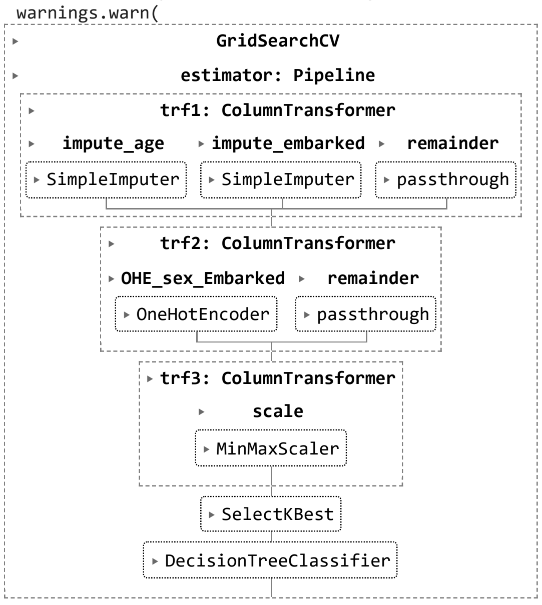
trf5 -> name of our model

```
from sklearn.model_selection import GridSearchCV
```

```
Grid = GridSearchCV(Pipe, params, cv=5, scoring='accuracy')
```

```
Grid.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
  warnings.warn(
```



just fitted grid on X_train and y_train

```
Grid.best_score_
```

```
0.6391214419383433
```

Gives best parameters (best depth)

Exporting Pipeline

```
import pickle
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
directory_path = '/content/drive/My Drive/Colab Notebooks/'
```

```
pickle.dump(Pipe, open(directory_path + 'Pipe.pkl', 'wb'))
```

All the techniques i.e., Simple Imputer, One Hot Encoding, etc are inside Pipe, So I don't need to add more while using the Pipeline.