Building ML Pipeline


```
# Importing Libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

```
churn_df = pd.read_csv('Churn_Modelling.csv')
```

```
churn_df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1

Next steps: Generate code with churn_df | View recommended plots | New interactive sheet

```
# Dropping the unwanted columns
churn_df.drop(columns = ['RowNumber','CustomerId','Surname'], inplace = True)
```

```
churn_df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	Female	42	2	0.00	1	1
1	608	Spain	Female	41	1	83807.86	1	0
2	502	France	Female	42	8	159660.80	3	1
3	699	France	Female	39	1	0.00	2	0
4	850	Spain	Female	43	2	125510.82	1	1

Next steps: Generate code with churn_df | View recommended plots | New interactive sheet

```
# Dividing the dataset into input features and target features

X = churn_df.drop(columns = ['Exited'])
y = churn_df['Exited']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

print(f'Count of rows in Training set : {X_train.shape[0]}')
print(f'Count of rows in Testing set : {X_test.shape[0]}')
```

```
Count of rows in Training set : 8000
Count of rows in Testing set : 2000
```

```
# Pipeline for processing numerical data

num_pipeline = Pipeline([
    ('num_imputation', SimpleImputer(strategy = 'mean')),
    ('feature_scaling', MinMaxScaler()),
    ('pca', PCA(0.98))
])

num_pipeline
```

```
Pipeline
SimpleImputer
MinMaxScaler
PCA
```

```
# Pipeline for processing categorical data

catg_pipeline = Pipeline([
    ('catg_imputation', SimpleImputer(fill_value = 'missing', strategy= 'constant')),
    ('one_hot_encoding', OneHotEncoder(sparse= False, handle_unknown = 'ignore'))
])

catg_pipeline
```

```
Pipeline
SimpleImputer
OneHotEncoder
```

```
num_cols = X.select_dtypes(include = np.number).columns.tolist()
catg_cols = X.select_dtypes(include = 'object').columns.tolist()
```

```
churn_df.info()
```

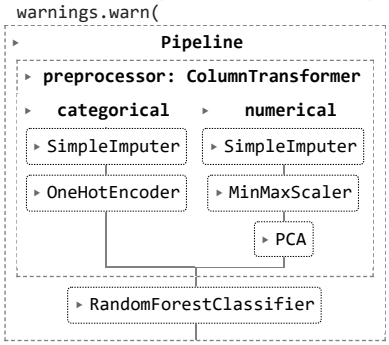
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   CreditScore      10000 non-null  int64
 1   Geography        10000 non-null  object
 2   Gender           10000 non-null  object
 3   Age              10000 non-null  int64
 4   Tenure           10000 non-null  int64
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64
 7   HasCrCard        10000 non-null  int64
 8   IsActiveMember   10000 non-null  int64
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

```
preprocessor = ColumnTransformer([
    ('categorical', catg_pipeline, catg_cols),
    ('numerical', num_pipeline, num_cols)
])

preprocessor
```

```
ColumnTransformer
categorical
SimpleImputer
OneHotEncoder
numerical
SimpleImputer
MinMaxScaler
PCA
```

```
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier())
])

pipe.fit(X_train, y_train)
```

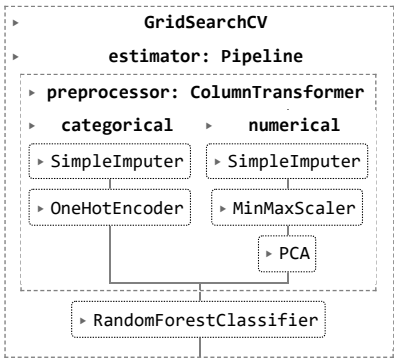
```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outp
  warnings.warn(
```




```
pipe.score(X_test, y_test) * 100
```

```
86.25
```

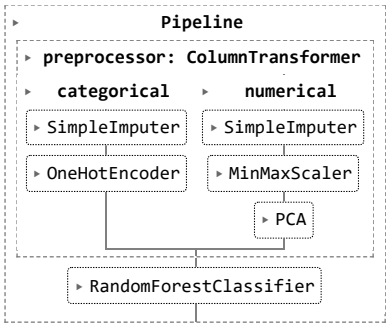
```
import warnings
warnings.filterwarnings('ignore')


# Hyperparameter tuning

parameters = {
    'classifier__n_estimators' : [100, 150, 200],
    'classifier__max_depth' : [5, 7, 10, 15],
    'classifier__min_samples_split' : [2, 3, 4],
    'classifier__max_features' : [2, 4, 6, 8, 10]
}

grid_search = GridSearchCV(
    pipe,
    param_grid= parameters,
    n_jobs=1
)

grid_search.fit(X_train, y_train)
```




```
grid_search.best_params_
```

```
{'classifier__max_depth': 10,
 'classifier__max_features': 6,
 'classifier__min_samples_split': 3,
 'classifier__n_estimators': 150}
```

```
pipe2 = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators = 100,
                                          max_features = 6,
                                          max_depth = 10,
                                          min_samples_split = 3))
])

pipe2.fit(X_train, y_train)
```



Start coding or generate with AI.