In [2]:
```
import pandas as pd

df = pd.read_csv('titanic.csv')

df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [7]:
```
X = df[['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']]
y= df['Survived']
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y)
```

Data Analysis

In [9]:
```
len(X)
```

Out[9]:
```
891
```

In [10]:
```
X.shape
```

Out[10]:
```
(891, 7)
```

In [14]:
```
X.isnull().sum() / len(X)*100
```

Out[14]:
```
Pclass       0.000000
Sex          0.000000
Age         19.865320
SibSp        0.000000
Parch        0.000000
Fare         0.000000
Embarked     0.224467
dtype: float64
```

In [15]:
```
X.dtypes
```

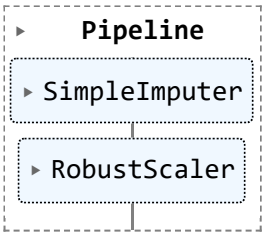
Out[15]:
```
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

Preprocessing

Numerical

In [18]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import RobustScaler

numerical_pipe = Pipeline([
    ('imputer',SimpleImputer(strategy='mean')),
    ('scaler', RobustScaler())
])

numerical_pipe
```

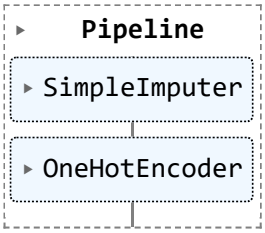
Out[18]:



Categorical

In [40]:
```
from sklearn.preprocessing import OneHotEncoder

cat_pipe = Pipeline([
    ('imputer',SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(sparse=False,
                              drop='if_binary',
                             handle_unknown = 'ignore'))
])

cat_pipe
```

Out[40]:



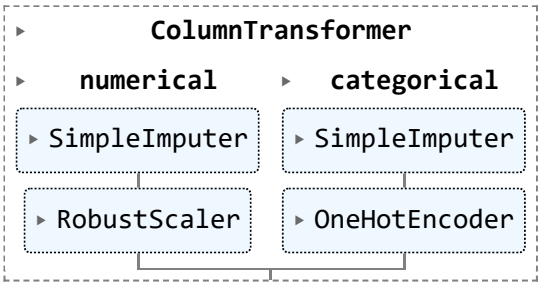
In [41]:
```
cat_pipe.fit_transform(X_train[['Embarked']])
```

Out[41]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       ...,
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.]])
```

Combined preprocessing pipeline

In [43]:
```
from sklearn.compose import ColumnTransformer
from sklearn.compose import make_column_selector

num_selector = make_column_selector(dtype_include=['float','int'])
cat_selector = make_column_selector(dtype_include=['object'])

preprocessing_pipe = ColumnTransformer([
    ('numerical', numerical_pipe, num_selector),
    ('categorical', cat_pipe, cat_selector)
])

preprocessing_pipe
```

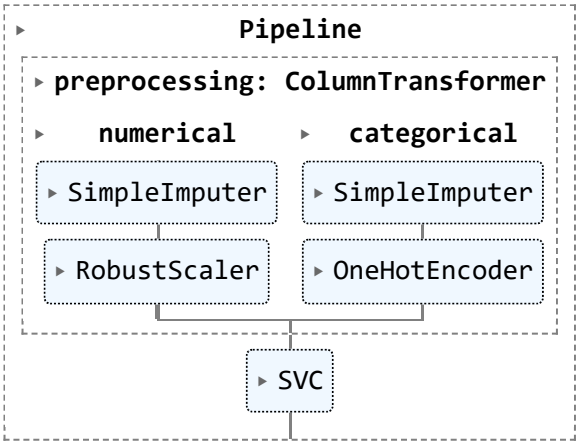
Out[43]:



Modelling Pipeline

In [44]:
```
from sklearn.svm import SVC

model_pipe = Pipeline([
    ('preprocessing', preprocessing_pipe),
    ('model', SVC())
])

model_pipe
```

Out[44]:



Evaluate and Tune

Baseline

In [46]:
```
from sklearn.model_selection import cross_validate

cv_results = cross_validate(model_pipe, X_train, y_train, cv=10)

cv_results
```

Out[46]:
```
{'fit_time': array([0.05246592, 0.03935194, 0.03527904, 0.03678107, 0.03490615,
        0.04382896, 0.03993988, 0.03641105, 0.03358006, 0.03271079]),
 'score_time': array([0.01870489, 0.01101708, 0.01181102, 0.01085472, 0.01206994,
        0.01458597, 0.01288891, 0.01131606, 0.01280308, 0.0177002 ]),
 'test_score': array([0.88059701, 0.85074627, 0.76119403, 0.79104478, 0.86567164,
        0.79104478, 0.74626866, 0.76119403, 0.83333333, 0.81818182])}
```

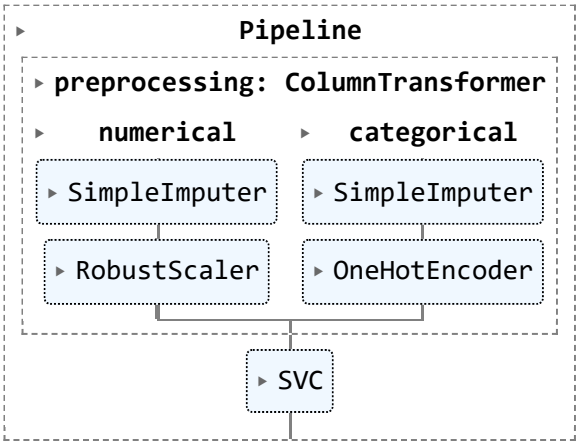
In [48]:
```
cv_results['test_score'].mean()
```

Out[48]:
```
0.8099276345545002
```

Tune

In [50]:
```
model_pipe
```

Out[50]:



In [49]:
```
model_pipe.get_params()
```

Out[49]:
```
{'memory': None,
 'steps': [('preprocessing',
   ColumnTransformer(transformers=[('numerical',
                                    Pipeline(steps=[('imputer', SimpleImputer()),
                                                    ('scaler', RobustScaler())]),
                                    <sklearn.compose._column_transformer.make_colu
mn_selector object at 0x1359ef880>),
                                   ('categorical',
                                    Pipeline(steps=[('imputer',
                                                     SimpleImputer(strategy='most_
frequent')),
                                                    ('encoder',
                                                     OneHotEncoder(drop='if_binar
y',
                                                                   handle_unknown
='ignore',
                                                                   sparse=Fals
e))]),
                                    <sklearn.compose._column_transformer.make_colu
mn_selector object at 0x1359ef7f0>)])),
  ('model', SVC())],
 'verbose': False,
 'preprocessing': ColumnTransformer(transformers=[('numerical',
                                   Pipeline(steps=[('imputer', SimpleImputer()),
                                                   ('scaler', RobustScaler())]),
                                   <sklearn.compose._column_transformer.make_column
_selector object at 0x1359ef880>),
                                  ('categorical',
                                   Pipeline(steps=[('imputer',
                                                    SimpleImputer(strategy='most_fr
equent')),
                                                   ('encoder',
                                                    OneHotEncoder(drop='if_binary',
                                                                  handle_unknown='i
gnore',
                                                                  sparse=False))]),
                                   <sklearn.compose._column_transformer.make_column
_selector object at 0x1359ef7f0>)]),
 'model': SVC(),
 'preprocessing__n_jobs': None,
 'preprocessing__remainder': 'drop',
 'preprocessing__sparse_threshold': 0.3,
 'preprocessing__transformer_weights': None,
 'preprocessing__transformers': [('numerical',
   Pipeline(steps=[('imputer', SimpleImputer()), ('scaler', RobustScaler())]),
   <sklearn.compose._column_transformer.make_column_selector at 0x1359ef880>),
  ('categorical',
   Pipeline(steps=[('imputer', SimpleImputer(strategy='most_frequent')),
                   ('encoder',
                    OneHotEncoder(drop='if_binary', handle_unknown='ignore',
                                  sparse=False))]),
   <sklearn.compose._column_transformer.make_column_selector at 0x1359ef7f0>)],
 'preprocessing__verbose': False,
 'preprocessing__verbose_feature_names_out': True,
 'preprocessing__numerical': Pipeline(steps=[('imputer', SimpleImputer()), ('scale
r', RobustScaler())]),
 'preprocessing__categorical': Pipeline(steps=[('imputer', SimpleImputer(strategy
='most_frequent')),
                 ('encoder',
                  OneHotEncoder(drop='if_binary', handle_unknown='ignore',
                                sparse=False))]),
 'preprocessing__numerical__memory': None,
 'preprocessing__numerical__steps': [('imputer', SimpleImputer()),
  ('scaler', RobustScaler())],
```

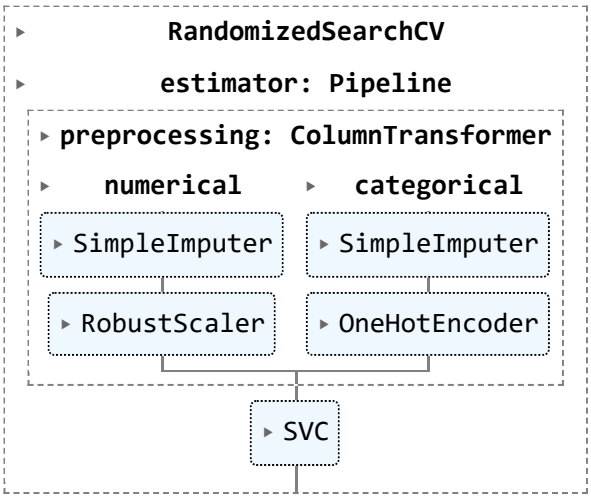
```
 'preprocessing__numerical__verbose': False,
 'preprocessing__numerical__imputer': SimpleImputer(),
 'preprocessing__numerical__scaler': RobustScaler(),
 'preprocessing__numerical__imputer__add_indicator': False,
 'preprocessing__numerical__imputer__copy': True,
 'preprocessing__numerical__imputer__fill_value': None,
 'preprocessing__numerical__imputer__missing_values': nan,
 'preprocessing__numerical__imputer__strategy': 'mean',
 'preprocessing__numerical__imputer__verbose': 'deprecated',
 'preprocessing__numerical__scaler__copy': True,
 'preprocessing__numerical__scaler__quantile_range': (25.0, 75.0),
 'preprocessing__numerical__scaler__unit_variance': False,
 'preprocessing__numerical__scaler__with_centering': True,
 'preprocessing__numerical__scaler__with_scaling': True,
 'preprocessing__categorical__memory': None,
 'preprocessing__categorical__steps': [('imputer',
   SimpleImputer(strategy='most_frequent')),
  ('encoder',
   OneHotEncoder(drop='if_binary', handle_unknown='ignore', sparse=False))],
 'preprocessing__categorical__verbose': False,
 'preprocessing__categorical__imputer': SimpleImputer(strategy='most_frequent'),
 'preprocessing__categorical__encoder': OneHotEncoder(drop='if_binary', handle_unk
nown='ignore', sparse=False),
 'preprocessing__categorical__imputer__add_indicator': False,
 'preprocessing__categorical__imputer__copy': True,
 'preprocessing__categorical__imputer__fill_value': None,
 'preprocessing__categorical__imputer__missing_values': nan,
 'preprocessing__categorical__imputer__strategy': 'most_frequent',
 'preprocessing__categorical__imputer__verbose': 'deprecated',
 'preprocessing__categorical__encoder__categories': 'auto',
 'preprocessing__categorical__encoder__drop': 'if_binary',
 'preprocessing__categorical__encoder__dtype': numpy.float64,
 'preprocessing__categorical__encoder__handle_unknown': 'ignore',
 'preprocessing__categorical__encoder__max_categories': None,
 'preprocessing__categorical__encoder__min_frequency': None,
 'preprocessing__categorical__encoder__sparse': False,
 'model__C': 1.0,
 'model__break_ties': False,
 'model__cache_size': 200,
 'model__class_weight': None,
 'model__coef0': 0.0,
 'model__decision_function_shape': 'ovr',
 'model__degree': 3,
 'model__gamma': 'scale',
 'model__kernel': 'rbf',
 'model__max_iter': -1,
 'model__probability': False,
 'model__random_state': None,
 'model__shrinking': True,
 'model__tol': 0.001,
 'model__verbose': False}
```

In [54]:

```
from sklearn.model_selection import RandomizedSearchCV
from scipy import stats

# Hyperparameter Grid
grid = {'model__C': stats.uniform(0.01, 10),
        'model__degree': stats.randint(1, 3),
        'model__kernel': ['linear','poly','rbf','sigmoid']}

# Instantiate Grid Search
search = RandomizedSearchCV(
    model_pipe,
    grid,
```

```
    scoring='accuracy',
    n_iter=100,  # number of draws
    cv=5,
    n_jobs=-1
)

# Fit data to Random Search
search.fit(X_train, y_train)
```

Out[54]:



In [55]:
```
search.best_params_
```

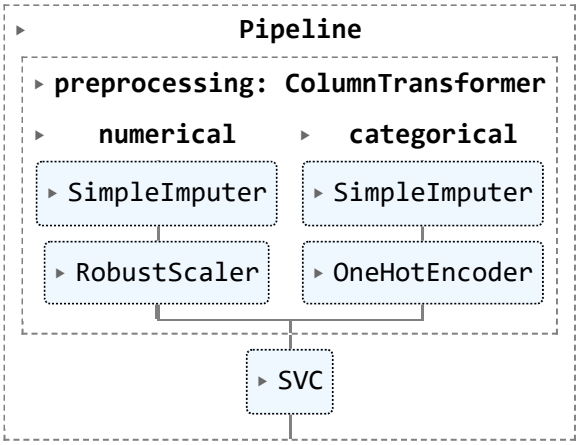
Out[55]:
```
{'model__C': 9.015140798612807, 'model__degree': 1, 'model__kernel': 'rbf'}
```

In [56]:
```
search.best_score_
```

Out[56]:
```
0.8113567500841656
```

In [57]:
```
best_model = search.best_estimator_
best_model
```

Out[57]:



Final test

In [58]:
```
best_model.score(X_test,y_test)
```

Out[58]:
```
0.8026905829596412
```

Export

In [59]:
```
import dill as pickle

# Export Pipeline as pickle file
with open("pipeline.pkl", "wb") as file:
    pickle.dump(best_model, file)
```