```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor


train_df = pd.read_csv("/content/train.csv")
test_df = pd.read_csv("/content/test.csv")


print(train_df.columns)
print(test_df.columns)
```

```
Index(['ID', 'Company', 'Quarter', 'QuickRatio', 'InventoryRatio',
       'RevenueGrowth', 'MarketshareChange', 'Bond rating', 'Stock rating',
       'Region', 'Industry', 'Sales'],
      dtype='object')
Index(['ID', 'Company', 'Quarter', 'QuickRatio', 'InventoryRatio',
       'RevenueGrowth', 'MarketshareChange', 'Bond rating', 'Stock rating',
       'Region', 'Industry'],
      dtype='object')
```

```
train_df = train_df.drop(["ID","Quarter"], axis=1)


train_df.dropna(subset=['Sales'], inplace=True)


# Splitting the data into features and target
X = train_df.drop(columns=["Sales"])
y= train_df["Sales"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

```
# Identify categorical columns (replace these with your actual column names)
categorical_cols = ['Bond rating', 'Stock rating', 'Region', 'Industry', 'Company

# Define the preprocessor as a ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        # Numerical transformer: Applies mean imputation and scaling
        ('num_transformer', Pipeline([
            ('num_imputer', SimpleImputer(strategy="mean")),
            ('scaler', StandardScaler())
        ]), ['InventoryRatio']),

        # Categorical transformer: Applies one-hot encoding to categorical column
        ('cat_transformer', OneHotEncoder(), categorical_cols)
    ],
    remainder='passthrough'  # Pass through any unspecified columns without trans
)
```

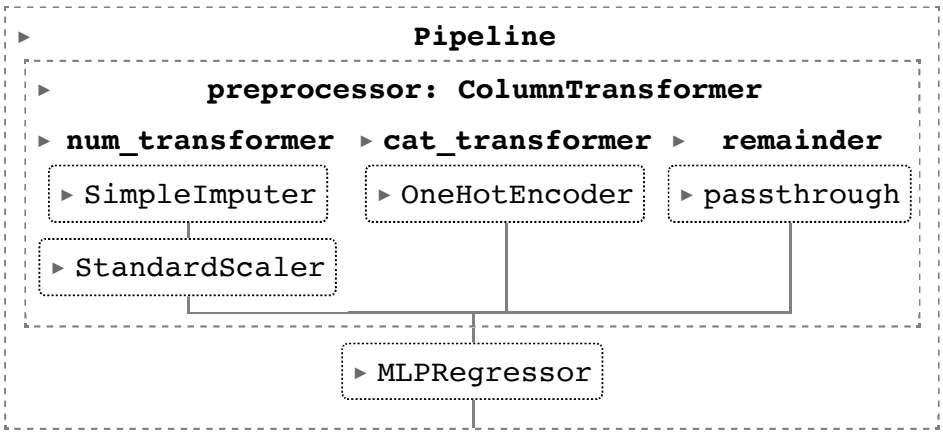
```
testing = preprocessor.fit_transform(X_train)
testing
```

```
<420x98 sparse matrix of type '<class 'numpy.float64'>'
        with 3578 stored elements in Compressed Sparse Row format>
```

```
# Define the MLPRegressor model
mlp_regressor = MLPRegressor(
    hidden_layer_sizes=(150, 100, 50),  # Architecture of the neural network
    activation='relu',  # Activation function
    solver='adam',  # Optimizer
    max_iter=1000 )
```

```
# Create a machine learning pipeline with the preprocessor and a regressor
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', mlp_regressor)
])
```

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




```
train_score = pipeline.score(X_train, y_train)
test_score = pipeline.score(X_test, y_test)

print(f"Training R^2 score: {train_score:.2f}")
print(f"Testing R^2 score: {test_score:.2f}")
```

```
Training R^2 score: 0.78
Testing R^2 score: 0.73
```

```
df_test = test_df["ID"]
```

```
test_df = test_df.drop(["ID","Quarter"], axis=1)
```

```
predictions = pipeline.predict(test_df)
```

```
predictions = pd.DataFrame(predictions,columns=["Sales"])
result = pd.concat([df_test,predictions],axis=1)
result.to_csv("submission2.csv",index=False)
```

```
submission = pd.read_csv("/content/submission2.csv")
submission
```

	ID	Sales
0	7	3056.726732
1	8	2416.266891
2	16	5381.977390
3	17	4495.726227
4	25	5070.298311
...	...	...
145	656	6081.478054
146	664	2971.420340
147	665	3572.239948
148	673	2223.053028
149	674	2873.313601

150 rows × 2 columns