In [1]:
```
import pandas as pd
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer , SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, KBinsDiscretize
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline, TransformerMixin
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

In [2]:
```
df = pd.read_csv('/Users/zanderbonnet/Desktop/GCU/DCS_530/Week 5/housing.csv')
df.head()
```

Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [3]:
```
X = df.drop('median_house_value',axis = 1)
y = df['median_house_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [4]:
```
numeric_transformer = Pipeline(steps=[
       ('imputer', IterativeImputer(random_state= 0))
      , ('discrete', KBinsDiscretizer(n_bins=25, encode='ordinal', strategy='kme
      ,('scaler', StandardScaler())
])
categorical_transformer = Pipeline(steps=[
       ('imputer', SimpleImputer(strategy='most_frequent'))
      ,('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

In [5]:
```
numeric_features = ['longitude', 'latitude', 'housing_median_age', 'total_rooms'
                    'households','median_income']
categorical_features = ['ocean_proximity']
preprocessor = ColumnTransformer(
   transformers=[
    ('numeric', numeric_transformer, numeric_features)
   ,('categorical', categorical_transformer, categorical_features)
])
```

In [6]:
```
pipeline = Pipeline(steps = [
               ('preprocessor', preprocessor)
```

```
            ,('regressor',LinearRegression())
        ])
```

In [7]:
```
mod = pipeline.fit(X_train,y_train)
mod.score(X_train,y_train)
```

Out[7]:
```
0.6616241542598114
```

In [8]:
```
mod.score(X_test,y_test)
```

Out[8]:
```
0.6482553668670538
```

In [9]:
```
pred = mod.predict(X_test)
r2_score(pred,y_test)
```

Out[9]:
```
0.46551312769346154
```

In [10]:
```
mean_squared_error(pred,y_test)**.5
```

Out[10]:
```
67724.37808770326
```