Project-2

USING LINEAR REGRATION

```
import pandas as pd

housing = pd.read_csv("housing.csv")

housing.head()

import matplotlib.pyplot as plt

housing.hist(bins=50, figsize=(10, 8))

plt.show()

from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)

import numpy as np

housing['income_cat'] = pd.cut(housing['median_income'], bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
labels=[1, 2, 3, 4, 5])

housing['income_cat'].hist()

plt.show()

from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)

for train_index, test_index in split.split(housing, housing["income_cat"]):

    strat_train_set = housing.loc[train_index]

    strat_test_set = housing.loc[test_index]

print(strat_test_set['income_cat'].value_counts() / len(strat_test_set)

for set_ in (strat_train_set, strat_test_set):

    set_.drop('income_cat', axis=1, inplace=True)

housing = strat_train_set.copy()

housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.4, s=housing['population']/100,
label='population',

figsize=(12, 8), c='median_house_value', cmap=plt.get_cmap('jet'), colorbar=True)
```

```
plt.legend()

plt.show()

corr_matrix = housing.corr()

print(corr_matrix.median_house_value.sort_values(ascending=False))

housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]

housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]

housing["population_per_household"] = housing["population"]/housing["households"]


corr_matrix = housing.corr()

print(corr_matrix["median_house_value"].sort_values(ascending=False))

# Data Preparation
                    housing = strat_train_set.drop("median_house_value", axis=1)
                    housing_labels = strat_train_set["median_house_value"].copy()
                    median = housing["total_bedrooms"].median()
                    housing["total_bedrooms"].fillna(median, inplace=True)
                    housing_num = housing.drop("ocean_proximity", axis=1)
                    from sklearn.base import BaseEstimator, TransformerMixin
                    # column index
                    rooms_ix, bedrooms_ix, population_ix, households_ix = 3, 4, 5, 6
                    class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
                        def __init__(self, add_bedrooms_per_room=True): # no *args or **kargs
                            self.add_bedrooms_per_room = add_bedrooms_per_room
                        def fit(self, X, y=None):
                            return self  # nothing else to do
                        def transform(self, X):
                            rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
                            population_per_household = X[:, population_ix] / X[:, households_ix]
                            if self.add_bedrooms_per_room:
                                bedrooms_per_room = X[:, bedrooms_ix] / X[:, rooms_ix]
                                return np.c_[X, rooms_per_household, population_per_household,
                                             bedrooms_per_room]
                            else:
                                return np.c_[X, rooms_per_household, population_per_household]
from sklearn.preprocessing import
OneHotEncoder
                                        from sklearn.pipeline import Pipeline
                                        from sklearn.preprocessing import StandardScaler
                                        from sklearn.impute import SimpleImputer
                                        num_pipeline = Pipeline([
```

```
    ('imputer',SimpleImputer(strategy="median")),
    ('attribs_adder', CombinedAttributesAdder()),
    ('std_scaler', StandardScaler()),
])
housing_num_tr = num_pipeline.fit_transform(housing_num)
from sklearn.compose import ColumnTransformer
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]
full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(), cat_attribs),
])
housing_prepared = full_pipeline.fit_transform(housing)
```

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(housing_prepared, housing_labels)


data = housing.iloc[:5]

labels = housing_labels.iloc[:5]

data_preparation = full_pipeline.transform(data)

print("Predictions: ", lin_reg.predict(data_preparation))
```