Project 1 - Part3. k-NN

In [1]:
```
# Load libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.model_selection import GridSearchCV

from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier

from dmba import regressionSummary
```

In [2]:
```
# Load data

df = pd.read_csv('FinalNew_house.csv')
df.columns
```

Out[2]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floor
s',
       'waterfront', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'zipcode', 'lat', 'long', 'sqft_living15',
       'sqft_lot15', 'age', 'renov_age'],
      dtype='object')
```

In [3]:
```
# Columns to be used were identified on previous section.

X = df[['grade', 'lat','age', 'sqft_living15']]
y = df['price']
```

In [4]:
```
# Split dataset into train and test

train_X, valid_X, train_y, valid_y = train_test_split(X,y, test_size=0.4
```

In [5]:
```
# Transform pandas dataframe to numpy array.

train_X_array = train_X.to_numpy()
valid_X_array = valid_X.to_numpy()
train_y_array = train_y.to_numpy()
valid_y_array = valid_y.to_numpy()
```

In [6]:
```
# Create standardizer
standardizer = StandardScaler()
```

In [7]:
```
# Standardize features
train_X_std = standardizer.fit_transform(train_X_array)
valid_X_std = standardizer.fit_transform(valid_X_array)
```

In [8]:
```
# Train a KNN classifier with 5 neighbors
knn5 = KNeighborsClassifier(n_neighbors=5, n_jobs=-1).fit(train_X_std, tı
knn5
```

Out[8]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_jobs=-1)
```

In [9]:
```
# Predict the class of two observations
y_predict_5 = knn5.predict(valid_X_std)
y_predict_5
```

Out[9]:
```
array([257000., 130000., 315450., ..., 235000., 445000., 185000.])
```

Identifying the Best Neighborhood Size

In [10]:
```
# Load library

from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.model_selection import GridSearchCV
```

In [11]:
```
# Create a KNN classifier
knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
```

In [12]:
```
# Create a pipeline
pipe = Pipeline([("standardizer", standardizer), ("knn", knn)])
```

In [13]:
```
# Create space of candidate values
search_space = [{"knn__n_neighbors": [ 1,2,3, 4, 5, 6, 7, 8, 9, 10,11,12,
```

In [14]:
```
# Create grid search
classifier = GridSearchCV(pipe, search_space, cv=5, verbose=0).fit(train_
```

```
C:\Users\daarv\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:684: UserWarning: The least populated class in y has only 1 members, whic
h is less than n_splits=5.
  warnings.warn(
```

In [15]:
```
# Best neighborhood size (k)
best_K = classifier.best_estimator_.get_params()["knn__n_neighbors"]
print('The best K for this K-NN model is: ', best_K)
```

```
The best K for this K-NN model is:  2
```

Model Statistics with k = 5

In [16]:
```
regressionSummary(valid_y_array, y_predict_5)
```

```
Regression statistics

                      Mean Error (ME) : 84916.5426
       Root Mean Squared Error (RMSE) : 141722.7023
            Mean Absolute Error (MAE) : 102478.5349
          Mean Percentage Error (MPE) : 15.7483
Mean Absolute Percentage Error (MAPE) : 21.5093
```

Model Statistics with k = 2

In [17]:
```
knn = KNeighborsClassifier(n_neighbors=2, n_jobs=-1).fit(train_X_std, tra

y_predicted = knn.predict(valid_X_std)
```

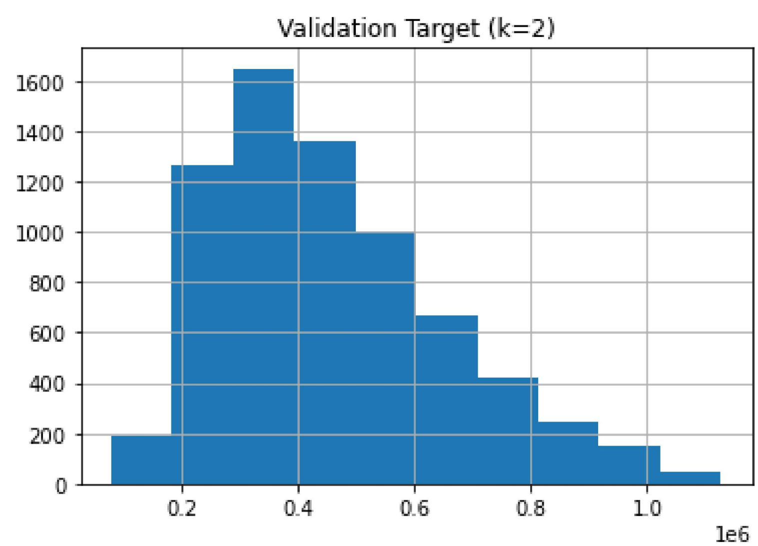
In [18]:
```
regressionSummary(valid_y_array, y_predicted)
```

```
Regression statistics

                      Mean Error (ME) : 43693.3936
       Root Mean Squared Error (RMSE) : 123017.5274
            Mean Absolute Error (MAE) : 85991.5124
          Mean Percentage Error (MPE) : 6.3275
Mean Absolute Percentage Error (MAPE) : 18.7976
```

In [19]:
```
pd.DataFrame(valid_y_array).hist()
plt.title('Validation Target (k=2)')
```

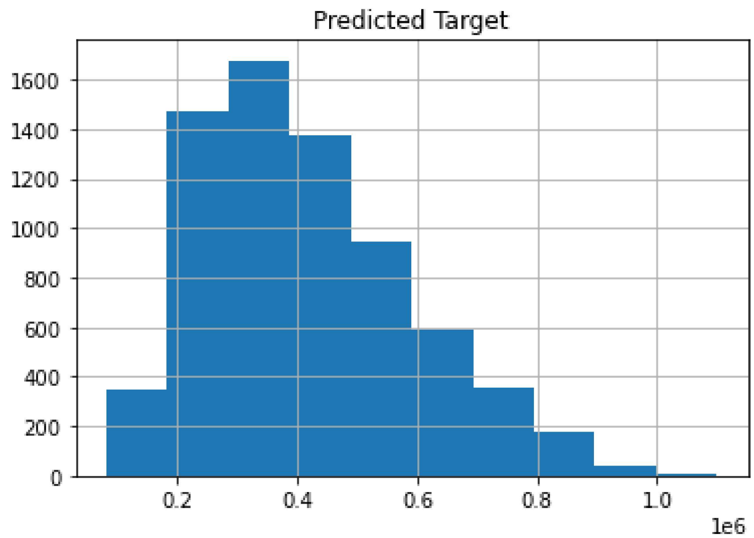
Out[19]: Text(0.5, 1.0, 'Validation Target (k=2)')



In [20]:

```
pd.DataFrame(y_predicted).hist()
plt.title('Predicted Target')
```

Out[20]: Text(0.5, 1.0, 'Predicted Target')



In [21]:

```
#fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=1, ncols=3)
fig, (ax1,ax2, ax3) = plt.subplots(nrows=1, ncols=3)

ax1.hist(y_predict_5, density=True, histtype='bar',color= 'red', stacked=
ax1.set_title('Predicted Target (k=5)')

ax2.hist(valid_y_array, histtype='bar', color= 'blue')
ax2.set_title('Validation Target')

ax3.hist(y_predicted, histtype='bar',color= 'tan')
ax3.set_title('Predicted Target (k=2)')

fig.tight_layout()
plt.show()
```