In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

In [2]:
```
# Import data
df = pd.read_csv('/Users/jack/Documents/GitHub/geospatial-data-science

# Read dataset
coast = gpd.read_file('/Users/jack/Documents/GitHub/geospatial-data-sc
```

In [3]:
```
df
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
0	-122.46	37.74	51	1905	291	707	
1	-122.00	36.93	51	1616	374	608	
2	-122.47	37.75	51	2413	431	1095	
3	-118.30	34.19	51	1502	243	586	
4	-118.32	33.83	51	2399	516	1160	
...	...	...	...	...	...	...	
15262	-117.23	33.83	2	1424	251	681	
15263	-119.88	36.83	2	4055	735	1730	
15264	-115.80	33.26	2	96	18	30	
15265	-122.00	38.23	1	2062	343	872	
15266	-120.93	37.65	1	2254	328	402	

15267 rows × 9 columns

In [4]:
```
coast
```

Out[4]:

	FNODE_	TNODE_	LPOLY_	RPOLY_	LENGTH	COUNTY_	COUNTY_ID	LUCODE	STATE
0	297	291	1	24	15270.99572	113	12	2	CA

In [5]:
```
df.columns
```

Out[5]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

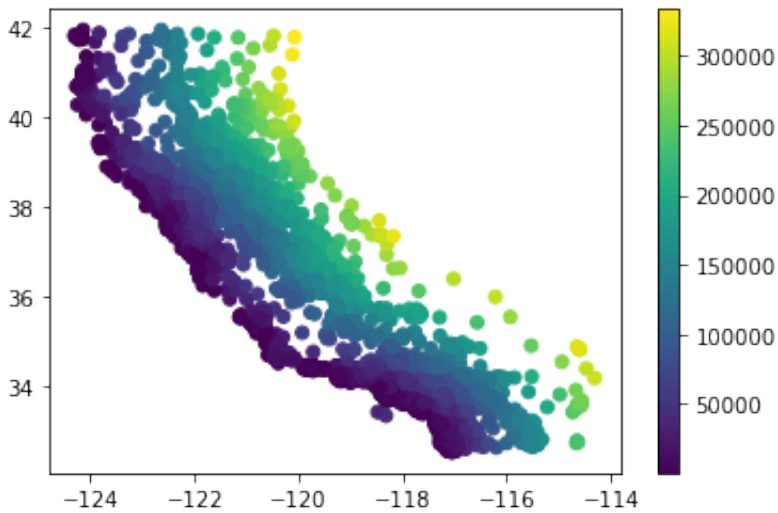
In [6]:
```
# Convert DataFrame to GeoDataFrame
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['longitude']
gdf = gdf.set_crs(4326, allow_override=True)

# Reproject everything to UTM 10N (EPSG:32610)
gdf_utm = gdf.to_crs('EPSG:32610')
coast_utm = coast.to_crs('EPSG:32610')
```

In [ ]:

In [7]:
```
# Compute distance to Space Needle
distance_to_coast = []
for i in range(gdf_utm.shape[0]):
    distance_to_coast.append(coast_utm.distance(gdf_utm['geometry'].il

# Add to DataFrame
gdf_utm['distance_to_coast'] = distance_to_coast
```

In [8]:
```
# Quickly check that it worked!
plt.scatter(gdf_utm['longitude'], gdf_utm['latitude'], c=gdf_utm['dist
plt.colorbar()
```

Out[8]:
```
<matplotlib.colorbar.Colorbar at 0x7ffc8257db50>
```



In [9]:
```
# Compute correlation matrix
corr_matrix = gdf_utm.corr()

# Display just house value correlations
corr_matrix["median_house_value"].sort_values(ascending= False)
```

Out[9]:
```
median_house_value    1.000000
median_income         0.668566
total_rooms           0.152923
households            0.098525
total_bedrooms        0.079023
population            0.020930
housing_median_age    0.014355
longitude            -0.020092
latitude             -0.173908
distance_to_coast    -0.505078
Name: median_house_value, dtype: float64
```

In [10]:
```
# Rooms per house
gdf_utm['rooms_per_house'] = gdf_utm['total_rooms'] / gdf_utm['househo

# Bedrooms per house
gdf_utm['bedrooms_per_room'] = gdf_utm['total_bedrooms'] / gdf_utm['to
```

In [11]:
```
# Compute correlation matrix
corr_matrix = gdf_utm.corr()

# Display just house value correlations
corr_matrix["median_house_value"].sort_values(ascending= False)
```

Out[11]:
```
median_house_value    1.000000
median_income         0.668566
total_rooms           0.152923
rooms_per_house       0.113277
households            0.098525
total_bedrooms        0.079023
population            0.020930
housing_median_age    0.014355
longitude            -0.020092
latitude             -0.173908
bedrooms_per_room    -0.233964
distance_to_coast    -0.505078
Name: median_house_value, dtype: float64
```

In [12]:
```
# Define feature list
feature_list =  ['median_income', 'distance_to_coast', 'bedrooms_per_r
                 'total_rooms', 'rooms_per_house', 'total_bedrooms', '

# Define features and labels
X = gdf_utm[feature_list]
y = gdf_utm['median_house_value']

# Standarize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [13]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_
```

In [14]:
```
# Define model
forest_reg = RandomForestRegressor(n_estimators = 30)

# Fit model
forest_reg.fit(X_train, y_train)
```

Out[14]:
```
RandomForestRegressor(n_estimators=30)
```

In [15]:
```
# Predict test labels predictions
predictions = forest_reg.predict(X_test)

# Compute mean-squared-error
final_mse = mean_squared_error(y_test , predictions)
final_rmse = np.sqrt(final_mse)
final_rmse
```

Out[15]:
```
55771.87429558656
```

Question 1

In [16]:
```
#from sklearn.ensemble import RandomForestRegressor

#forest_reg = RandomForestRegressor(n_estimators = 30)

#forest_reg.fit
```

In [17]:
```
port data for Seattle
tle_data = pd.read_csv('/Users/jack/Documents/GitHub/geospatial-data-s

ad dataset
tle_coast = gpd.read_file('/Users/jack/Documents/GitHub/geospatial-dat
```

In [18]:
```
seattle_data.columns
```

Out[18]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', '
yr_built',
       'lat', 'long'],
      dtype='object')
```

In [19]:
```
seattle_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19451 entries, 0 to 19450
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   price        19451 non-null  int64  
 1   bedrooms     19451 non-null  int64  
 2   bathrooms    19451 non-null  float64
 3   sqft_living  19451 non-null  int64  
 4   sqft_lot     19451 non-null  int64  
 5   yr_built     19451 non-null  int64  
 6   lat          19451 non-null  float64
 7   long         19451 non-null  float64
dtypes: float64(3), int64(5)
memory usage: 1.2 MB
```

In [20]:
```
seattle_coast_utm = seattle_coast.to_crs(32610)
```

In [21]:
```
seattle_coast_utm
```

Out[21]:

	NAME	MTFCC	geometry
0	Pacific	L4150	LINESTRING (505407.678 5323560.749, 505406.177...
1	Pacific	L4150	LINESTRING (544009.095 5240908.754, 544015.188...
2	Pacific	L4150	LINESTRING (524251.546 5362207.085, 524261.425...
3	Pacific	L4150	LINESTRING (543812.537 5345133.673, 543810.169...
4	Pacific	L4150	LINESTRING (420246.733 5124016.964, 419988.090...
...	...	...	...
165	Pacific	L4150	LINESTRING (424777.900 5081688.350, 424801.962...
166	Pacific	L4150	LINESTRING (443524.779 5118719.482, 443512.294...
167	Pacific	L4150	LINESTRING (447014.682 5121216.461, 447041.556...
168	Pacific	L4150	LINESTRING (447495.516 5121428.219, 447457.454...
169	Pacific	L4150	LINESTRING (448673.161 5121006.191, 448640.818...

170 rows × 3 columns

In [22]:
```
# Check the number of houses
seattle_data
```

Out[22]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	yr_built	lat	long
0	538000	3	2.25	2570	7242	1951	47.7210	-122.319
1	180000	2	1.00	770	10000	1933	47.7379	-122.233
2	604000	4	3.00	1960	5000	1965	47.5208	-122.393
3	510000	3	2.00	1680	8080	1987	47.6168	-122.045
4	1230000	4	4.50	5420	101930	2001	47.6561	-122.005
...	...	...	...	...	...	...	...	...
19446	475000	3	2.50	1310	1294	2008	47.5773	-122.409
19447	360000	3	2.50	1530	1131	2009	47.6993	-122.346
19448	400000	4	2.50	2310	5813	2014	47.5107	-122.362
19449	400000	3	2.50	1600	2388	2004	47.5345	-122.069
19450	325000	2	0.75	1020	1076	2008	47.5941	-122.299

19451 rows × 8 columns

In [23]:
```
seattle_data.isnull()
```

Out[23]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	yr_built	lat	long
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
19446	False	False	False	False	False	False	False	False
19447	False	False	False	False	False	False	False	False
19448	False	False	False	False	False	False	False	False
19449	False	False	False	False	False	False	False	False
19450	False	False	False	False	False	False	False	False

19451 rows × 8 columns

In [24]:
```
# Check the null value in dataset
print(seattle_data.isnull().sum())
```

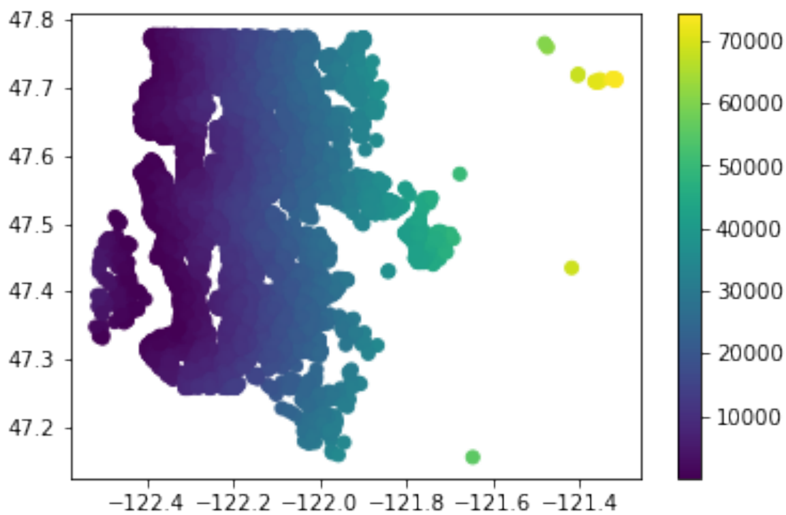
```
price          0
bedrooms       0
bathrooms      0
sqft_living    0
sqft_lot       0
yr_built       0
lat            0
long           0
dtype: int64
```

In [25]:
```
# Convert DataFrame to GeoDataFrame
seattle_gdf = gpd.GeoDataFrame(seattle_data, geometry=gpd.points_from_
seattle_gdf = seattle_gdf.set_crs(4326, allow_override=True)

# Reproject everything to UTM 10N (EPSG:32610)
seattle_gdf_utm = seattle_gdf.to_crs('EPSG:32610')
seattle_coast_utm = seattle_coast.to_crs('EPSG:32610')
```

In [26]:
```
# Compute distance to coast
seattle_coast = []
for i in range(seattle_gdf_utm.shape[0]):
    seattle_coast.append(seattle_coast_utm.distance(seattle_gdf_utm['g

# Add to DataFrame
seattle_gdf_utm['seattle_coast'] = seattle_coast
```

In [27]:
```
# Quickly check that it worked!
plt.scatter(seattle_gdf_utm['long'], seattle_gdf_utm['lat'], c=seattle
plt.colorbar()
```

Out[27]:
```
<matplotlib.colorbar.Colorbar at 0x7ffc863fd3a0>
```



In [28]:
```
gdf_utm.columns
```

Out[28]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'geometry', 'distance_to_coast',
       'rooms_per_house', 'bedrooms_per_room'],
      dtype='object')
```

In [29]:
```
seattle_gdf_utm.columns
```

Out[29]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', '
yr_built',
       'lat', 'long', 'geometry', 'seattle_coast'],
      dtype='object')
```

Question 2, 4-5

In [30]:
```
# Compute correlation matrix
corr_matrix = seattle_gdf_utm.corr()

# Display just house value correlations
corr_matrix["price"].sort_values(ascending= False)
```

Out[30]:
```
price           1.000000
sqft_living     0.702296
bathrooms       0.524395
bedrooms        0.315804
lat             0.308082
sqft_lot        0.090125
yr_built        0.052453
seattle_coast   0.027830
long            0.020092
Name: price, dtype: float64
```

Question 3

In [31]:
```
seattle_data.isnull()
```

Out[31]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	yr_built	lat	long	geometry
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
19446	False	False	False	False	False	False	False	False	False
19447	False	False	False	False	False	False	False	False	False
19448	False	False	False	False	False	False	False	False	False
19449	False	False	False	False	False	False	False	False	False
19450	False	False	False	False	False	False	False	False	False

19451 rows × 9 columns

In [32]:
```
# Import library
from sklearn.preprocessing import StandardScaler
# Define feature list
seattle_feature_list =  ['bedrooms', 'bathrooms', 'sqft_living', 'sqft
       'lat', 'long']

# Define features and labels
seattle_X = seattle_gdf_utm[seattle_feature_list]
seattle_y = seattle_gdf_utm['price']

# Standarize data
seattle_scaler = StandardScaler()
seattle_X_scaled = seattle_scaler.fit_transform(seattle_X)
```

In [33]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(seattle_X_scaled,
```

In [34]:
```
# Define model
forest_reg = RandomForestRegressor(n_estimators = 30)

# Fit model
forest_reg.fit(X_train, y_train)
```

Out[34]:
```
RandomForestRegressor(n_estimators=30)
```

In [35]:
```
# Predict test labels predictions
seattle_predictions = forest_reg.predict(X_test)

# Compute mean-squared-error
final_mse = mean_squared_error(y_test , seattle_predictions)
final_rmse = np.sqrt(final_mse)
final_rmse
```

Out[35]:
```
161998.9710966534
```

Check for correlation

In [36]:
```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

sea = pd.read_csv("/Users/jack/Documents/GitHub/geospatial-data-scienc
sea_gdf = gpd.GeoDataFrame(sea, geometry = gpd.points_from_xy(sea['lon
sea_gdf = sea_gdf.set_crs(4326, allow_override=True)
# Reproject everything to UTM 10N (EPSG:32610)
sea_utm = sea_gdf.to_crs('EPSG:32610')

rand_point = Point(5.6e5, 5.27e6)
rand_gdf = gpd.GeoDataFrame(geometry = [rand_point])
rand_gdf = rand_gdf.set_crs(32610)
dist_sea_rand_pnt = []
for i in range(sea_utm.shape[0]):
    dist_sea_rand_pnt.append(rand_gdf.distance(sea_utm['geometry'].ilo
```

Correlation matrix for square foot divide by bedroom

In [37]:
```
# House size
seattle_gdf_utm['house_size'] = seattle_gdf_utm['sqft_living'] / seatt
```

In [38]:
```
seattle_gdf_utm.columns
```

Out[38]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', '
yr_built',
       'lat', 'long', 'geometry', 'seattle_coast', 'house_size'],
      dtype='object')
```

In [39]:
```
# Import library
from sklearn.preprocessing import StandardScaler
# Define feature list
seattle_feature_list =  ['bedrooms', 'bathrooms', 'sqft_living', 'sqft
       'lat', 'long','house_size']

# Define features and labels
seattle_X = seattle_gdf_utm[seattle_feature_list]
seattle_y = seattle_gdf_utm['price']

# Standarize data
seattle_scaler = StandardScaler()
seattle_X_scaled = seattle_scaler.fit_transform(seattle_X)
```

In [40]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(seattle_X_scaled,

# Define model
forest_reg = RandomForestRegressor(n_estimators = 30)

# Fit model
forest_reg.fit(X_train, y_train)

# Predict test labels predictions
seattle_predictions = forest_reg.predict(X_test)

# Compute mean-squared-error
seattle_final_mse = mean_squared_error(y_test , seattle_predictions)
seattle_final_rmse = np.sqrt(seattle_final_mse)
seattle_final_rmse
```

Out[40]: 153023.53775454397

Compute distance to Washington Park Arboretum UW Seattle

In [41]:
```
#47.63982848173617, -122.29614021004966
from shapely.geometry import Point
uw_arboretum = Point( -122.28120001226782, 47.6167207495107)
uw_arboretum_gdf = gpd.GeoDataFrame(geometry = [uw_arboretum], crs = 4
uw_arboretum_gdf = uw_arboretum_gdf.to_crs("EPSG:32610")
distance_to_Arboretum = []
for i in range(seattle_gdf_utm.shape[0]):
    distance_to_Arboretum.append(uw_arboretum_gdf.distance(seattle_gdf
# Add to DataFrame
seattle_gdf_utm['distance_to_Arboretum'] = distance_to_Arboretum
```

In [42]:
```
seattle_gdf_utm.columns
```

Out[42]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', '
yr_built',
       'lat', 'long', 'geometry', 'seattle_coast', 'house_size',
       'distance_to_Arboretum'],
      dtype='object')
```

In [43]:
```
# Import library
from sklearn.preprocessing import StandardScaler
# Define feature list
seattle_feature_list =  ['bedrooms', 'bathrooms', 'sqft_living', 'sqft
       'lat', 'long','distance_to_Arboretum']

# Define features and labels
seattle_X = seattle_gdf_utm[seattle_feature_list]
seattle_y = seattle_gdf_utm['price']

# Standarize data
seattle_scaler = StandardScaler()
seattle_X_scaled = seattle_scaler.fit_transform(seattle_X)
```

In [44]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(seattle_X_scaled,
```

In [45]:
```
# Define model
forest_reg = RandomForestRegressor(n_estimators = 30)

# Fit model
forest_reg.fit(X_train, y_train)
```

Out[45]:
```
RandomForestRegressor(n_estimators=30)
```

In [46]:
```
# Predict test labels predictions
seattle_predictions = forest_reg.predict(X_test)

# Compute mean-squared-error
seattle_final_mse = mean_squared_error(y_test , seattle_predictions)
seattle_final_rmse = np.sqrt(seattle_final_mse)
seattle_final_rmse
```

Out[46]:
```
150907.06671017915
```

```
The houses near Washington Park Arboretum UW are more expensive, so
it's correlated with the house price.
```

Compute distance to Seattle University, WA

In [47]:
```
# 47.609774530443964, -122.3177216635164
from shapely.geometry import Point
seattle_University = Point(-122.3177216635164, 47.609774530443964)
seattle_University_gdf = gpd.GeoDataFrame(geometry = [seattle_Universi
seattle_University_gdf = seattle_University_gdf.to_crs("EPSG:32610")
distance_to_seattle_University = []
for i in range(seattle_gdf_utm.shape[0]):
    distance_to_seattle_University.append(seattle_University_gdf.dista
# Add to DataFrame
seattle_gdf_utm['distance_to_seattle_University'] = distance_to_seattl
```

In [48]:
```
seattle_gdf_utm.columns
```

Out[48]:
```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', '
yr_built',
       'lat', 'long', 'geometry', 'seattle_coast', 'house_size',
       'distance_to_Arboretum', 'distance_to_seattle_University'],
      dtype='object')
```

In [49]:
```
# Import library
from sklearn.preprocessing import StandardScaler
# Define feature list
seattle_feature_list =  ['bedrooms', 'bathrooms', 'sqft_living', 'sqft
       'lat', 'long', 'distance_to_seattle_University']

# Define features and labels
seattle_X = seattle_gdf_utm[seattle_feature_list]
seattle_y = seattle_gdf_utm['price']

# Standarize data
seattle_scaler = StandardScaler()
seattle_X_scaled = seattle_scaler.fit_transform(seattle_X)
```

In [50]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(seattle_X_scaled,
```

In [51]:
```
# Define model
forest_reg = RandomForestRegressor(n_estimators = 30)

# Fit model
forest_reg.fit(X_train, y_train)
```

Out[51]:
```
RandomForestRegressor(n_estimators=30)
```

In [52]:
```
# Predict test labels predictions
seattle_predictions = forest_reg.predict(X_test)

# Compute mean-squared-error
seattle_final_mse = mean_squared_error(y_test , seattle_predictions)
seattle_final_rmse = np.sqrt(seattle_final_mse)
seattle_final_rmse
```

Out[52]:
```
155020.88638338653
```

```
The compute distance to Washington Park Arboretum UW Seattle seems
more accurate than the distance to Seattle University.
```

Questions/Answers

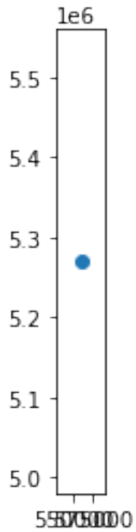
```
1. How many houses are in this dataset?
There are 19451 houses in this dataset.

2. How many features are there for predicting house price?
All 7 features, (bedrooms,bathrooms, sqft_living, sqft_lot, yr_built,
lat,long) can predict the house price because the number of bedroom,
bathroom, the area size, the age of the house and its location can
affect prices.

3. Are there any null values in this dataset?
There are no null values in this dataset.

4. Which three variables are best correlated with house price
(include correlation coefficients)?
The total house square footage of the house, bathroom and number of
bedrooms, are best correlated with house price.

5. Which three variables are least correlated with house price
(include correlation coefficients)?
The age of the house, the distance to coast and longitude, are least
correlated with house price.
```

Insang Demo

In [53]:
```
from shapely.geometry import Point
rand_point = Point(5.6e5, 5.27e6)
rand_gdf = gpd.GeoDataFrame(geometry = [rand_point])
```

In [54]:
```
rand_gdf.plot()
```

Out[54]: <AxesSubplot:>



In [ ]: