



CPE 232 Data Models Final Project

EDA and house value prediction model from California housing prices data

จัดทำโดย

นายชัชนันท์	บุญพา	65070501014
นายณัฐชนนท์	ปฐมานุรักษ์	65070501018
นายนพดล	หาญกิตติกาญจนา	65070501032
นายณวิน	โตศิลานนท์	65070501033
นายปรัตษกรณ์	กิตติชฎาพงศ์	65070501036

เสนอ

ผศ.ดร.สันติธรรม พรหมอ่อน

รายงานนี้เป็นส่วนหนึ่งของรายวิชา CPE232 Data Models

คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

ภาคเรียนที่ 2 ปีการศึกษา 2566

Introduction

รายงานโปรเจกต์รายวิชา CPE 232 Data model สำหรับการจัดทำ modeling และ visualization โดยใช้ชุดข้อมูล California Housing Prices จาก kaggle โดยชุดข้อมูลจะเป็นข้อมูลเกี่ยวกับบ้าน เช่น ราคา จำนวนห้อง และอื่นๆ ที่สำรวจในปี 1990 ใน ประเทศสหรัฐอเมริกา รัฐ California

Data explanation

ชุดข้อมูล California Housing Prices เป็นข้อมูลเกี่ยวข้องกับบ้านที่พบในเขต แคลิฟอร์เนีย และสถิติโดยสรุป โดยข้อมูลจะอ้างอิงจากการสำรวจสำมะโนประชากรปี 1990 ซึ่งจะแบ่งเป็น column ดังนี้

- ❖ **longitude** คือ หน่วยวัดพิกัดทางภูมิศาสตร์เพื่อบอกว่าที่อยู่ห่างจากทิศตะวันตกเท่าไร
- ❖ **latitude** คือ หน่วยวัดพิกัดทางภูมิศาสตร์เพื่อบอกว่าที่อยู่ห่างจากทิศเหนือเท่าไร
- ❖ **housing_median_age** คือ median (มัธยฐาน) อายุของบ้านในแต่ละบล็อก;
- ❖ **total_rooms** คือ จำนวนของห้องทั้งหมดภายใน 1 บล็อก
- ❖ **total_bedrooms** คือ ห้องนอนทั้งหมดภายใน 1 บล็อก
- ❖ **population** คือ จำนวนประชากรที่อาศัยอยู่ภายใน 1 บล็อก
- ❖ **households** คือ จำนวนกลุ่มของผู้อยู่อาศัยภายใน 1 บล็อก
- ❖ **median_income** คือ รายได้เฉลี่ยของครัวเรือนภายใน 1 บล็อก (หมื่นดอลลาร์สหรัฐ)
- ❖ **median_house_value** คือ ค่าเฉลี่ยของราคาบ้านที่อยู่ภายใน 1 บล็อก
- ❖ **ocean_proximity** คือ ตำแหน่งของบ้านโดยมีที่อยู่ของทะเลหรือมหาสมุทรเป็นการอ้างอิง

โดยข้อมูลมีการเผยแพร่ครั้งแรกในเอกสาร Pace, R. Kelley, and Ronald Barry. "Sparse spatial autoregressions." Statistics & Probability Letters 33.3 (1997): 291-297. และเป็นเวอร์ชันแก้ไขข้อมูล California Housing ที่หาได้จาก: หน้าของ Luís Torgo (มหาวิทยาลัยปอร์โต)

Data preparation process and results


❖ ดูภาพรวมข้อมูล

ดูภาพรวมของแต่ละ column ว่ามี data-type เป็นอะไรบ้าง จาก file csv โดยใช้ pandas.info และ pandas.head

```
[2] import pandas as pd

[6] df = pd.read_csv('housing.csv')

[ ] df.info()
```

 <class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):

#	Column	Non-Null Count	Dtype
0	longitude	20640 non-null	float64
1	latitude	20640 non-null	float64
2	housing_median_age	20640 non-null	float64
3	total_rooms	20640 non-null	float64
4	total_bedrooms	20433 non-null	float64
5	population	20640 non-null	float64
6	households	20640 non-null	float64
7	median_income	20640 non-null	float64
8	median_house_value	20640 non-null	float64
9	ocean_proximity	20640 non-null	object

dtypes: float64(9), object(1)
memory usage: 1.6+ MB

```
[7] df.head()
```



	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

❖ Cleaning Data

- เช็ค null value ของแต่ละ column โดยใช้ `pandas.isnull().any()` และจะเห็นได้ว่า `total_bedrooms` มี null อยู่

```
[ ] df.isnull().any()

longitude      False
latitude        False
housing_median_age  False
total_rooms     False
total_bedrooms   True
population      False
households      False
median_income   False
median_house_value False
ocean_proximity False
dtype: bool
```

```
[ ] df.fillna(0, inplace=True)
df.isnull().any()

longitude      False
latitude        False
housing_median_age  False
total_rooms     False
total_bedrooms   False
population      False
households      False
median_income   False
median_house_value False
ocean_proximity False
dtype: bool
```

ซึ่งเนื่องจากเรารู้ว่า data-type ของ `total_bedrooms` คือ float64 ดังนั้นจึงใช้ `fillna(0, inplace=True)` ใส่เลข 0 แทน null value ไป

- เช็คการกระจายตัวของข้อมูล โดยใช้ `describe()` ซึ่งจะอธิบายข้อมูลดังนี้
 - count คือ จำนวนของแถวในแต่ละ column
 - mean คือ ค่าเฉลี่ยของแถวทั้งหมดในแต่ละ column
 - std คือ ส่วนเบี่ยงเบนมาตรฐานของแถวทั้งหมดในแต่ละ column
 - min คือ ค่าต่ำสุดในแต่ละ column
 - 25% คือ เปอร์เซนไทล์ที่ 25 ของแต่ละ column
 - 50% คือ เปอร์เซนไทล์ที่ 50 ของแต่ละ column
 - 75% คือ เปอร์เซนไทล์ที่ 75 ของแต่ละ column
 - max คือ ค่าสูงสุดในแต่ละ column

```
[ ] df.describe()

count    longitude    latitude    housing_median_age    total_rooms    total_bedrooms    population    households    median_income    median_house_value
mean    -119.569704    35.631861    28.639486    2635.763081    532.476211    1425.476744    499.539680    3.870671    206855.816909
std       2.003532       2.135952       12.585558    2181.615252    422.678333    1132.462122    382.329753    1.899822    115395.615874
min      -124.350000    32.540000       1.000000       2.000000       0.000000       3.000000       1.000000    0.499900    14999.000000
25%      -121.800000    33.930000       18.000000    1447.750000    292.000000    787.000000    280.000000    2.563400    119600.000000
50%      -118.490000    34.260000       29.000000    2127.000000    431.000000    1166.000000    409.000000    3.534800    179700.000000
75%      -118.010000    37.710000       37.000000    3148.000000    643.250000    1725.000000    605.000000    4.743250    264725.000000
max      -114.310000    41.950000       52.000000    39320.000000    6445.000000    35682.000000    6082.000000    15.000100    500001.000000
```

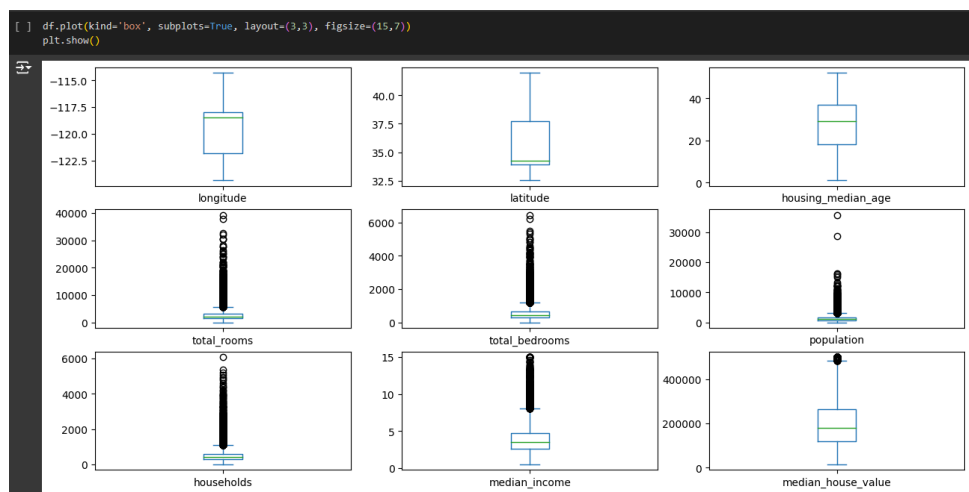
- นับจำนวน categorical data ที่อยู่ใน column ocean_proximity จะเห็นได้ว่าเกิด outlier ขึ้น คือ ISLAND เนื่องจากมีจำนวนที่น้อยกว่าอันอื่นมาก ดังนั้นจึงตัด ISLAND ใน column ocean_proximity ออก

```
[ ] df['ocean_proximity'].value_counts()

ocean_proximity
<1H OCEAN      9136
INLAND         6551
NEAR OCEAN     2658
NEAR BAY       2290
ISLAND           5
Name: count, dtype: int64
```

```
[ ] df = df[df['ocean_proximity'] != 'ISLAND']
```

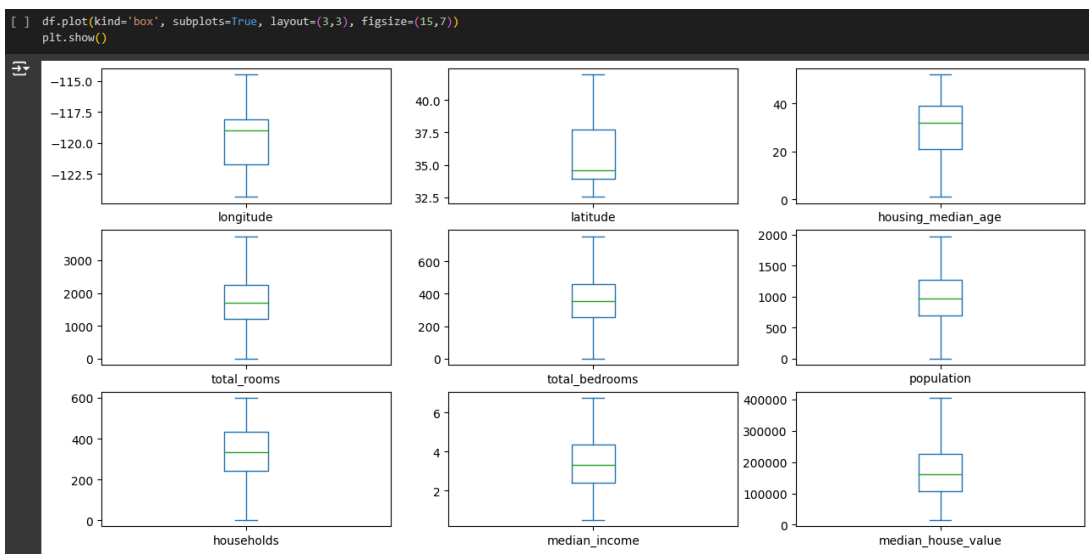
- นำข้อมูลทั้งหมดไปทำ box-plot เพื่อดู outlier ก็จะได้เห็นว่า total_rooms total_bedrooms population households และ median_income มี outlier เกิดขึ้น



- กำจัด outlier โดยการตัด quantile ในช่วงที่สูงออก

```
[ ] df=df[df['total_rooms']<df['total_rooms'].quantile(0.92)]
df=df[df['total_bedrooms']<df['total_bedrooms'].quantile(0.95)]
df=df[df['population']<df['population'].quantile(0.95)]
df=df[df['households']<df['households'].quantile(0.95)]
df=df[df['median_house_value']<df['median_house_value'].quantile(0.95)]
df=df[df['median_income']<df['median_income'].quantile(0.98)]
```

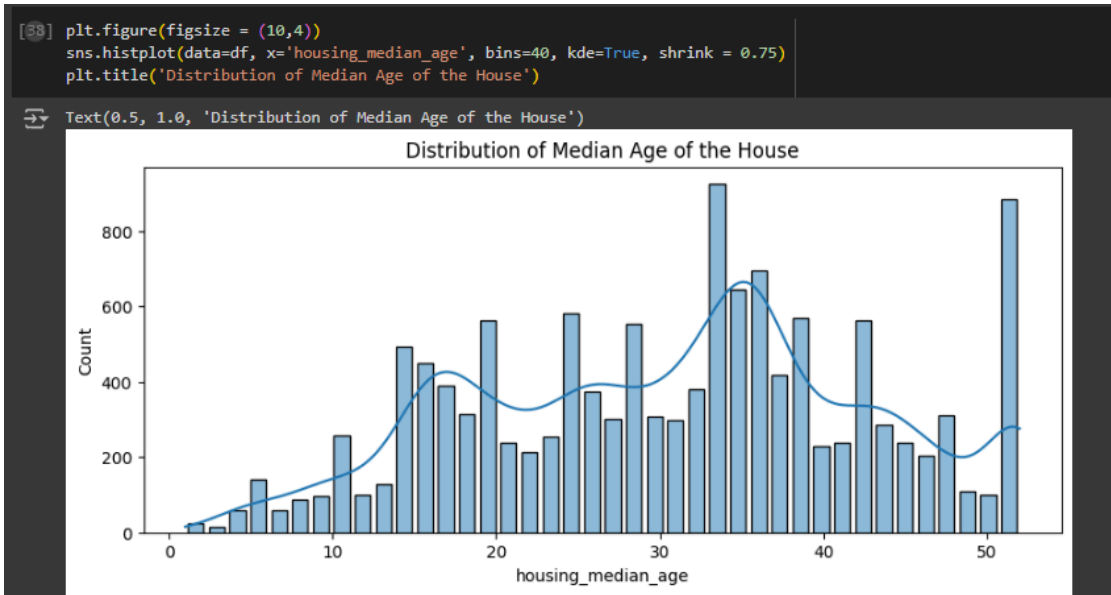
- ข้อมูลหลังจากที่ทำการกำจัด outlier



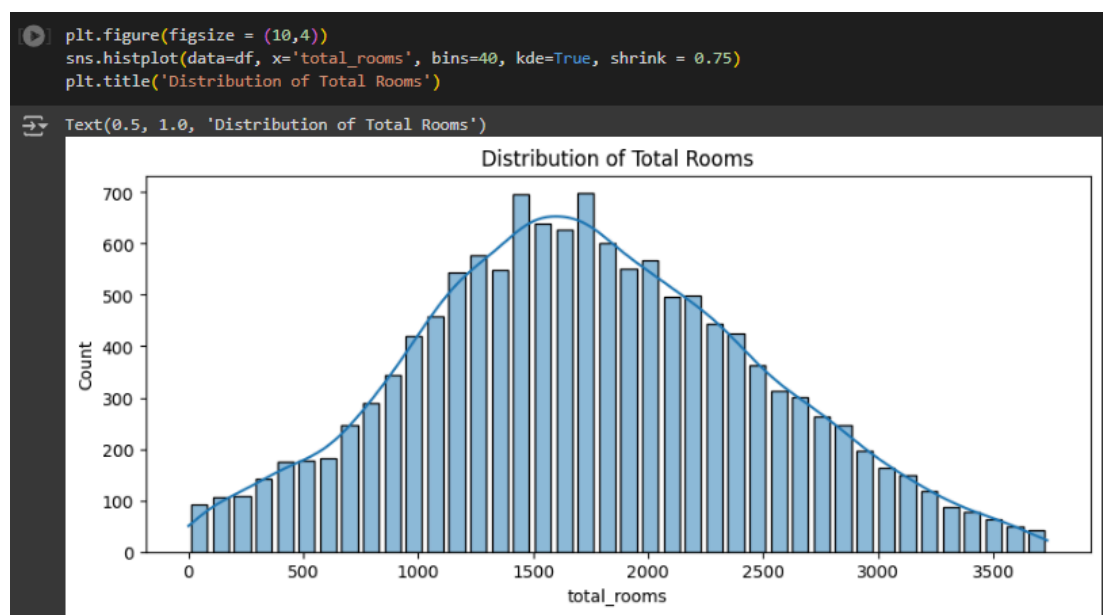
EDA and visualization of data

❖ Data distribution

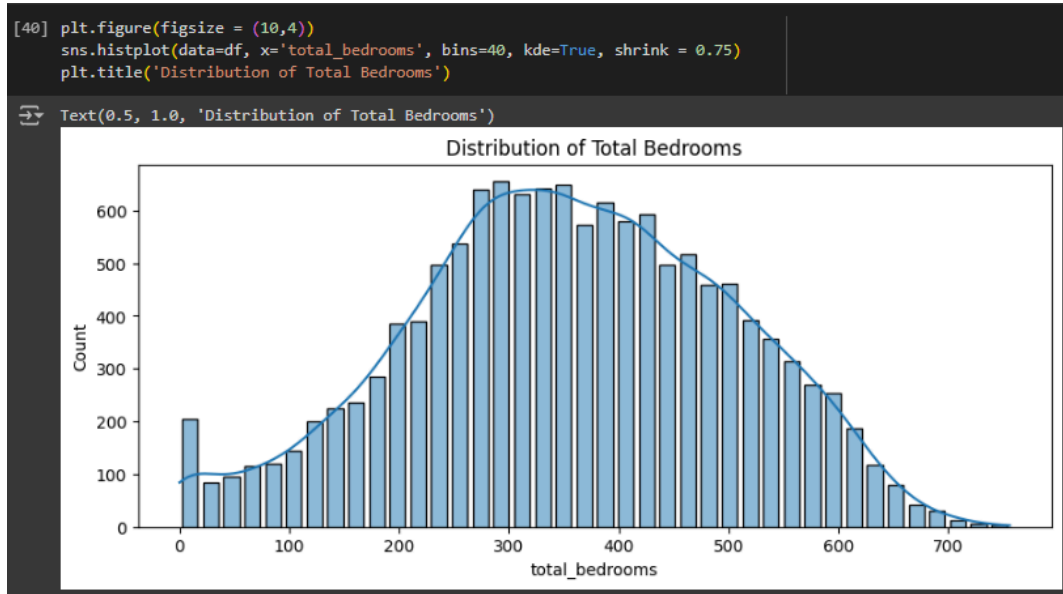
➤ Distribution ของมัธยฐานอายุของบ้าน



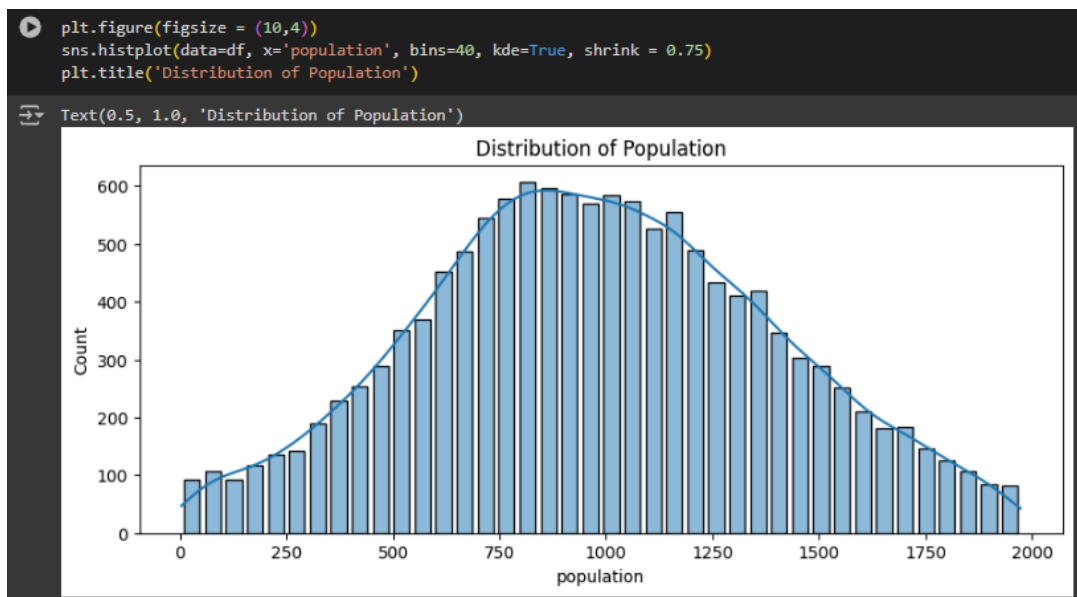
➤ Distribution ของจำนวนห้องใน 1 บล็อก ก็จะได้เห็นได้ว่าการกระจายตัวแบบ normal distribution การกระจายตัวสูงสุดจะอยู่ช่วง 1500 - 1700



- Distribution ของจำนวนห้องนอนใน 1 บล็อก มีการกระจายตัวแบบ normal distribution การกระจายตัวสูงสุดจะอยู่ในช่วง 300 - 400



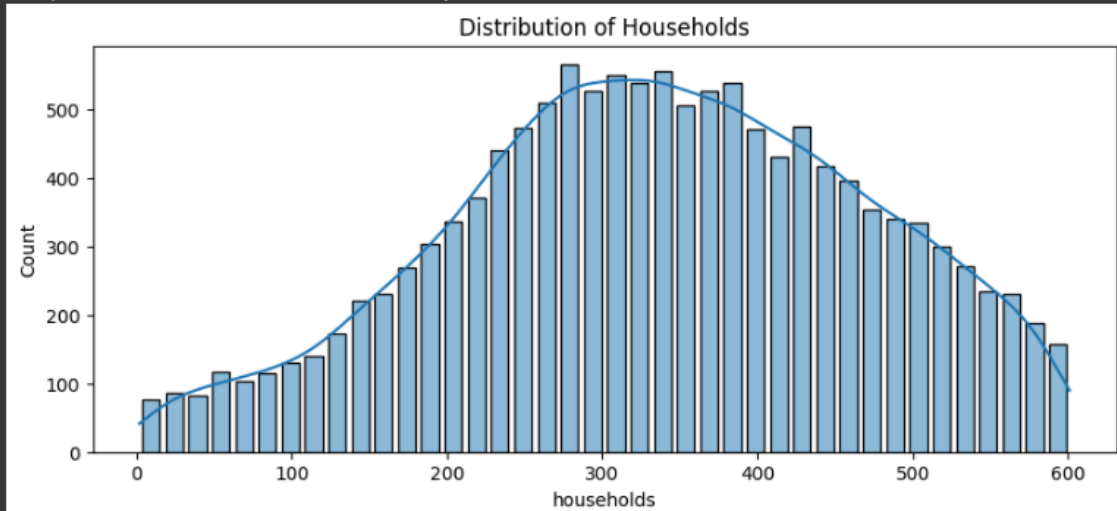
- Distribution ของจำนวนประชากรที่อาศัยอยู่ภายใน 1 บล็อก กระจายตัวแบบ normal distribution ช่วงที่มีการกระจายตัวสูงสุด คือ 750 - 1000



- Distribution ของจำนวนกลุ่มของผู้อยู่อาศัยภายใน 1 บล็อก

```
[42] plt.figure(figsize = (10,4))
sns.histplot(data=df, x='households', bins=40, kde=True, shrink = 0.75)
plt.title('Distribution of Households')
```

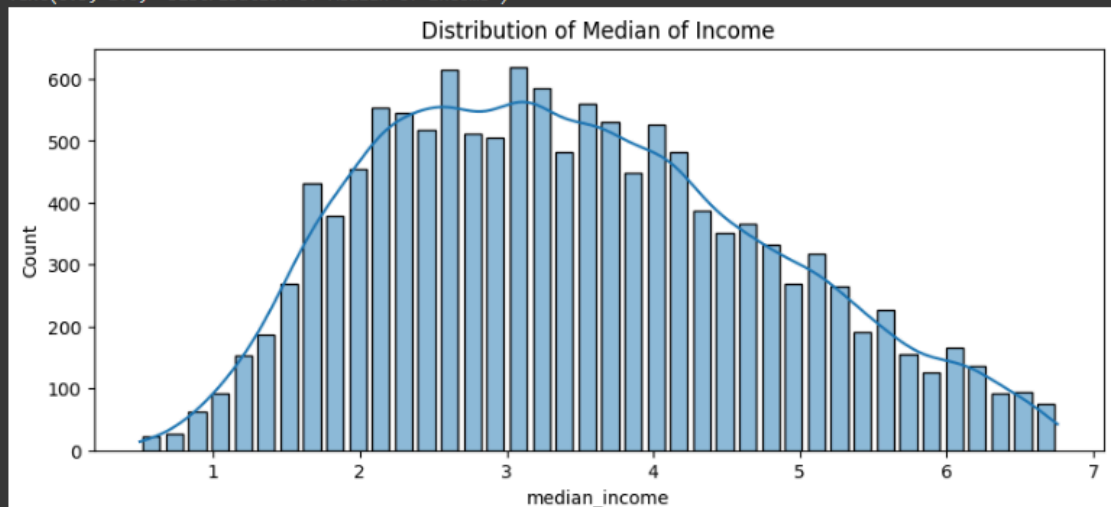
```
Text(0.5, 1.0, 'Distribution of Households')
```



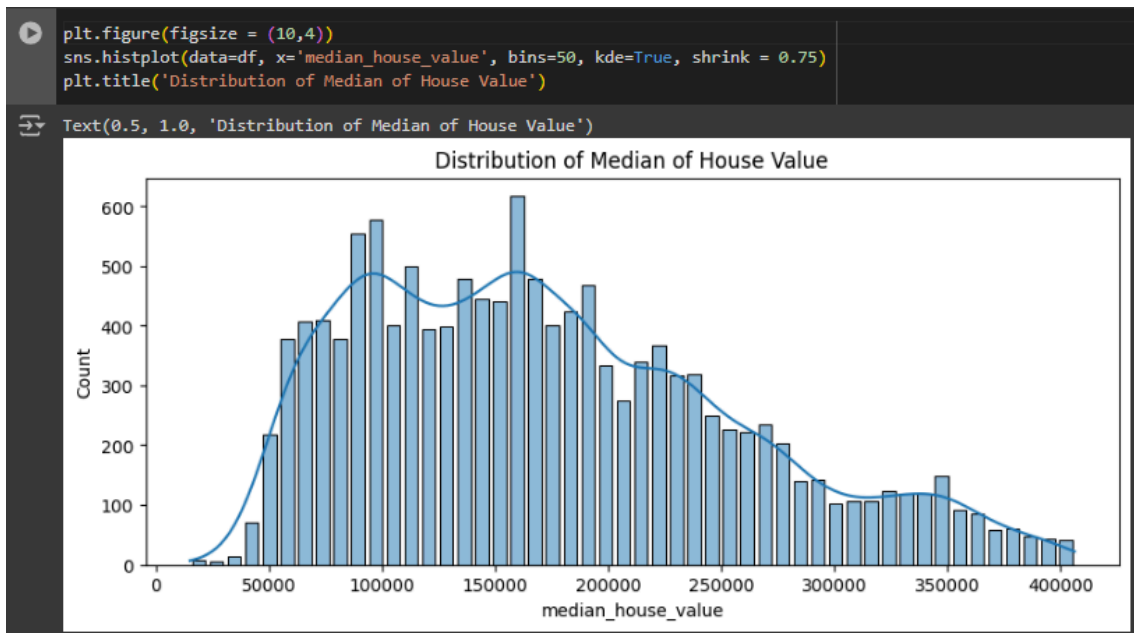
- Distribution ของรายได้เฉลี่ยของครัวเรือนภายใน 1 บล็อก (หมื่นดอลลาร์สหรัฐ) มีการกระจายตัวเป็น normal distribution

```
plt.figure(figsize = (10,4))
sns.histplot(data=df, x='median_income', bins=40, kde=True, shrink = 0.75)
plt.title('Distribution of Median of Income')
```

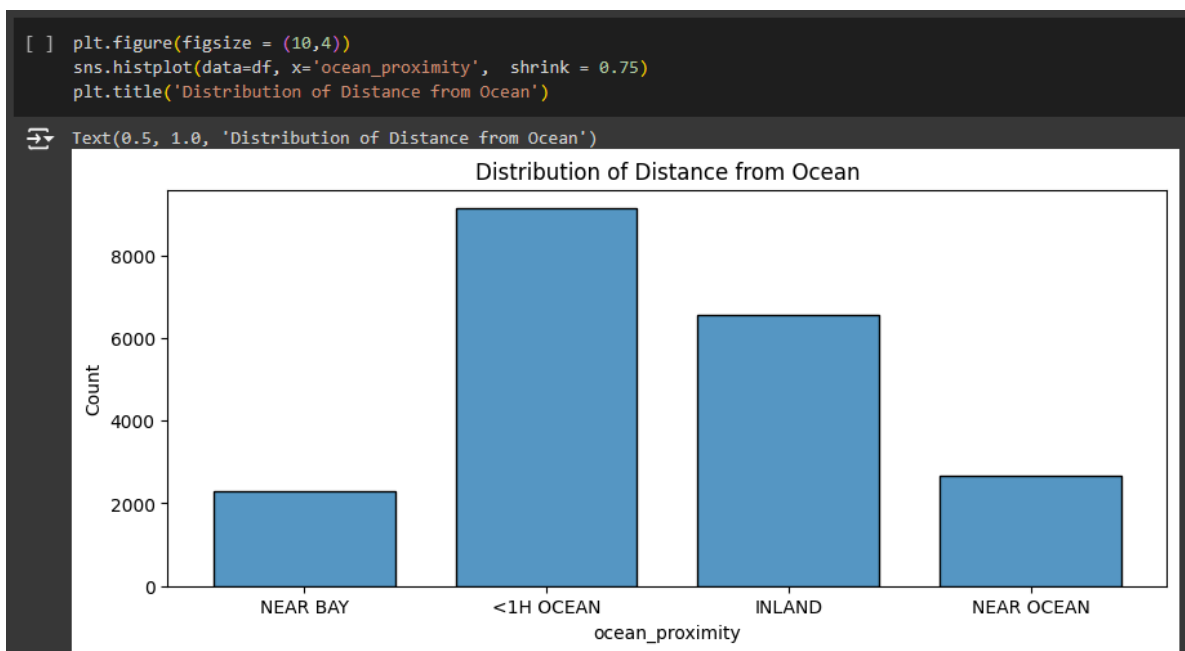
```
Text(0.5, 1.0, 'Distribution of Median of Income')
```



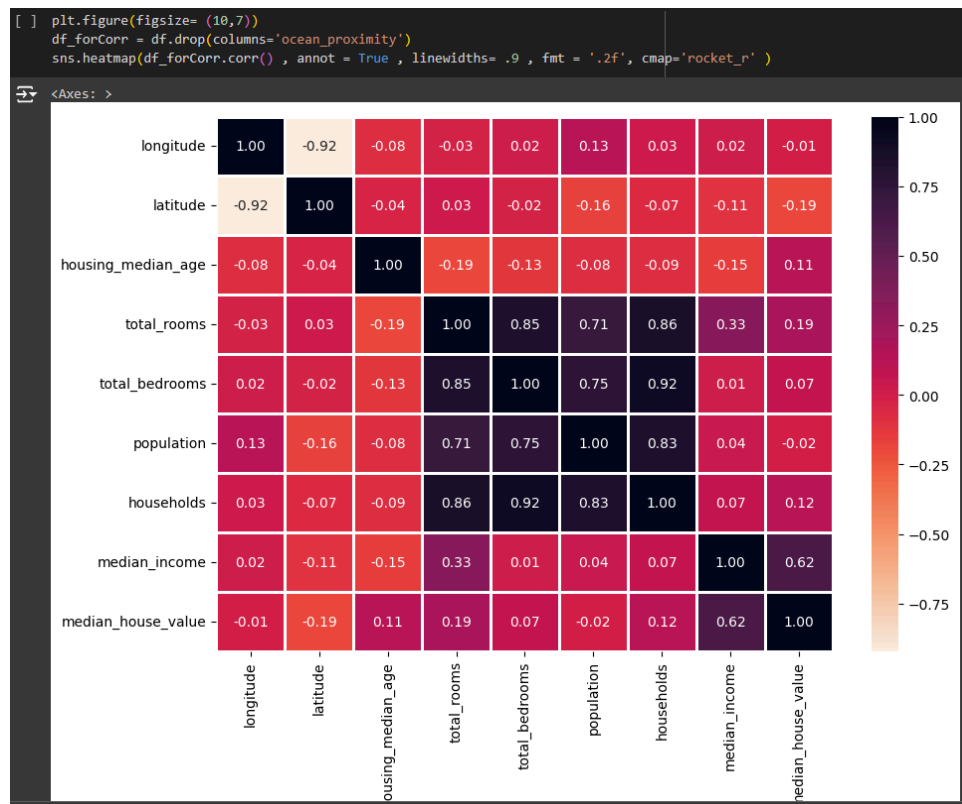
- Distribution ของค่าเฉลี่ยของราคาบ้านที่อยู่ภายใน 1 บล็อก



- Uniform Distribution ของตำแหน่งของบ้านโดยมีที่อยู่ของทะเลหรือมหาสมุทรเป็นการอ้างอิง

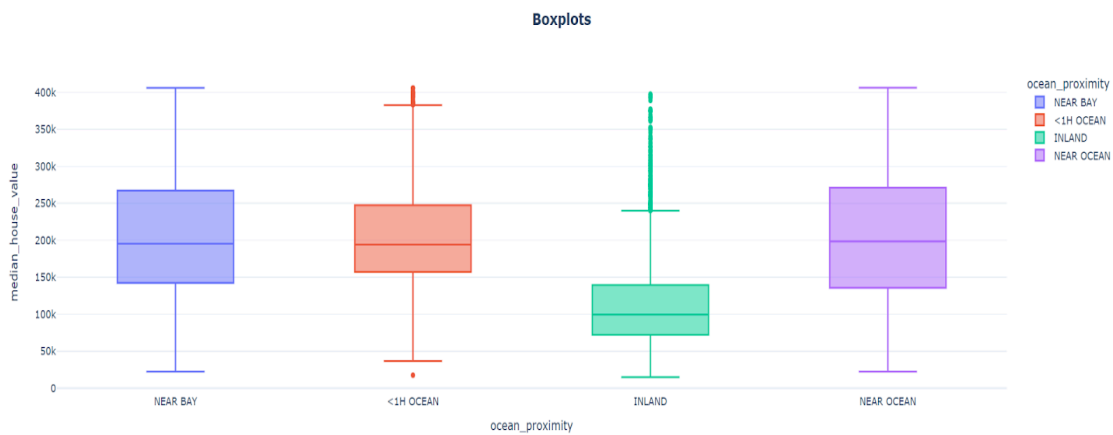


➤ สร้าง heat-map เพื่อดู correlation ของทุกตัวกับ ocean_proximity



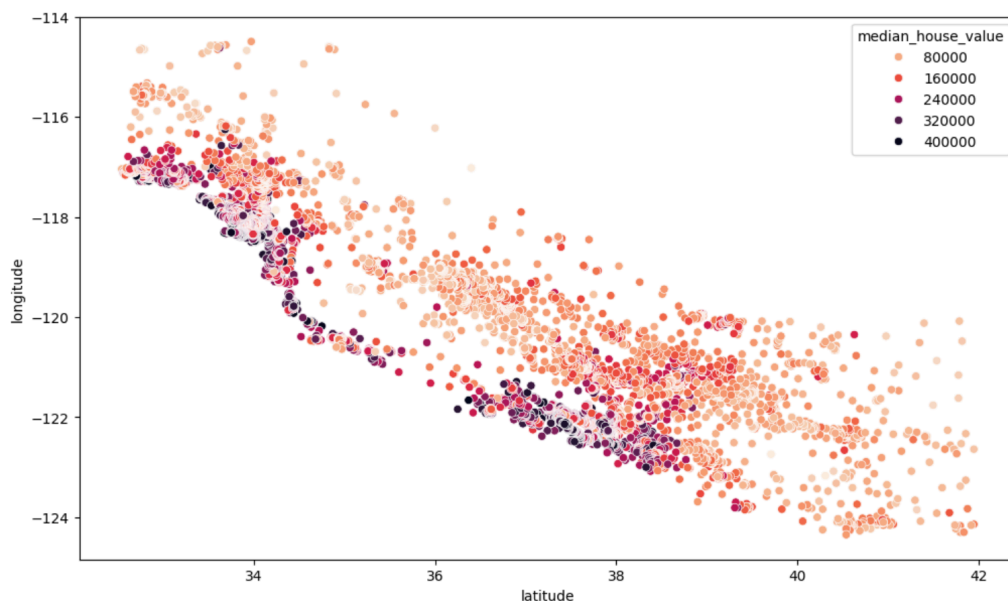
จาก heat-map จะเห็นว่า field ที่มีความสัมพันธ์กันจะเป็น total rooms total_bedroom population และ households ซึ่งเป็นสิ่งที่สมเหตุสมผลนั่นคือ จำนวนห้อง จำนวนห้องนอน จำนวนประชากรที่พักอาศัย และจำนวนครัวเรือนที่พักอาศัยมีความสัมพันธ์ไปในทางเดียวกันเป็นอย่างมาก และ median_house_value กับ median_income มีความสัมพันธ์ไปในทางเดียวกันเล็กน้อย

- Boxplots แสดงการกระจายตัวของราคาบ้านที่อยู่ภายใน 1 บล็อกของแต่ละตำแหน่งของบ้านกับมหาสมุทร



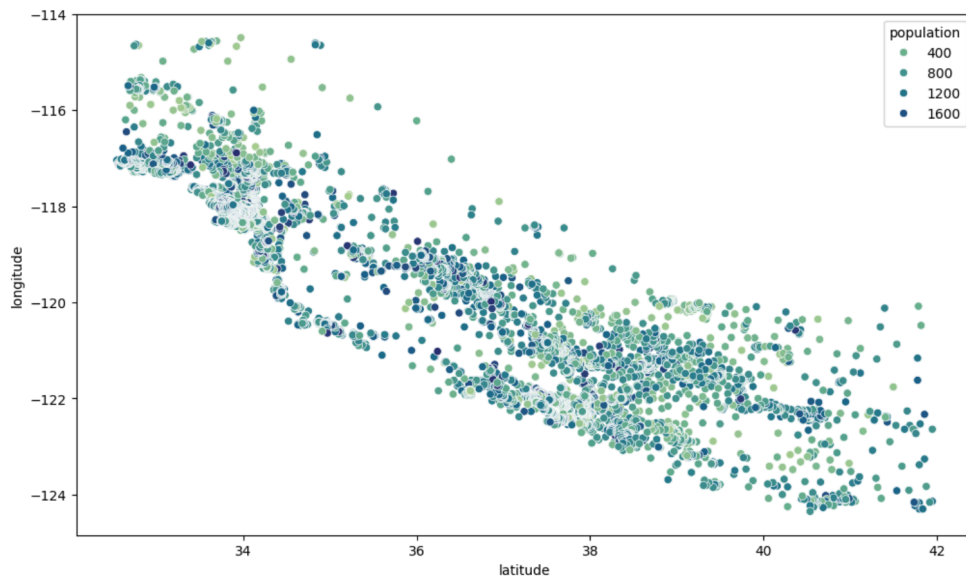
จะสังเกตเห็นได้ว่าบริเวณที่ไม่ได้อยู่ใกล้อ่าวหรือมหาสมุทรส่วนมากจะมีมูลค่าของบ้านต่ำกว่าบริเวณที่อยู่ใกล้อ่าวหรือมหาสมุทร

- Scatterplot แสดงตำแหน่งทางภูมิศาสตร์ที่ระบุด้วยละติจูดและลองจิจูด และสีของแต่ละจุดแสดงค่าเฉลี่ยของราคาบ้าน



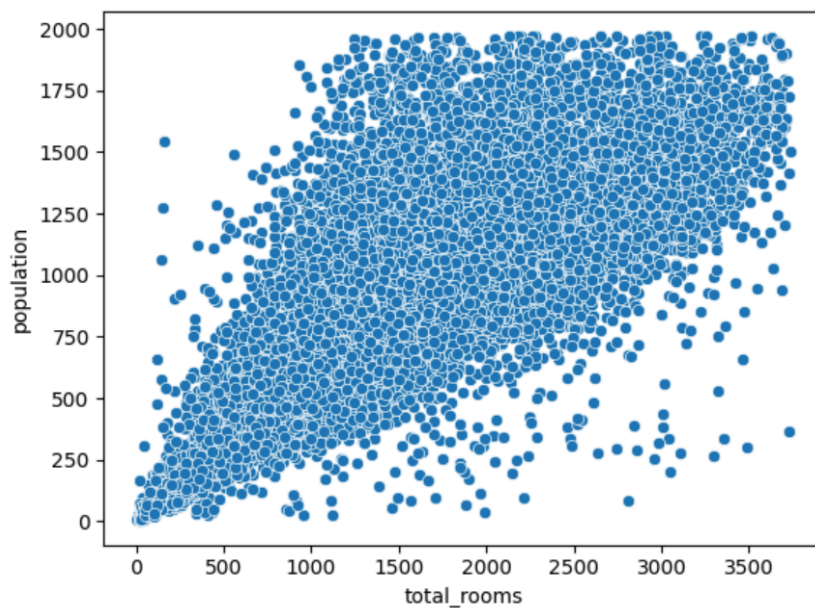
ผลลัพธ์ที่ได้มีลักษณะตามผลของ Boxplots ก่อนหน้าซึ่งแสดงให้เห็นว่าบริเวณที่อยู่ใกล้อ่าวหรือมหาสมุทรจะมีมูลค่าของบ้านสูงกว่า

- Scatterplot แสดงตำแหน่งทางภูมิศาสตร์ที่ระบุด้วยละติจูดและลองจิจูด และสีของแต่ละจุดแสดงจำนวนประชากร



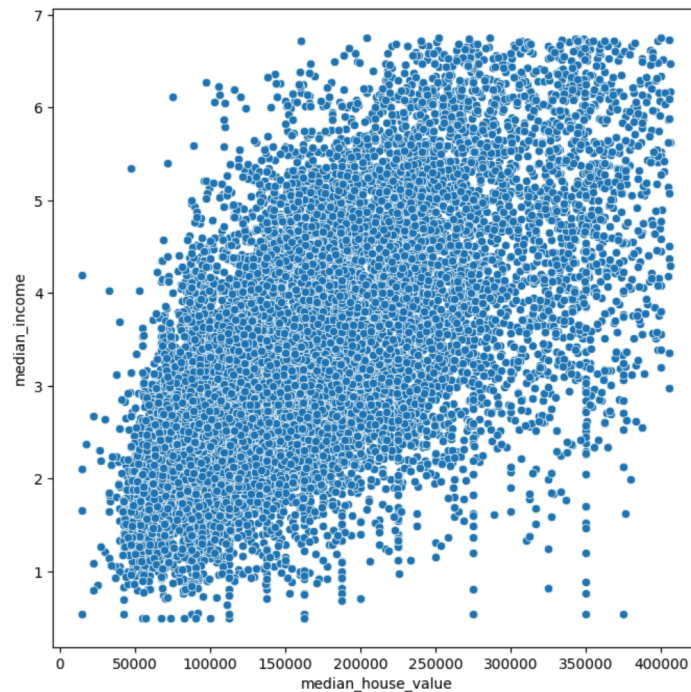
ผลลัพธ์ที่ได้แสดงให้เห็นว่าประชากรที่อาศัยอยู่มีการกระจายตัวทั่วทั้งรัฐ california แต่ก็มีจุดสังเกตที่บางบริเวณจะมีประชากรหนาแน่นกว่าบริเวณอื่นอยู่บ้าง

- Scatterplot ระหว่างจำนวนห้องทั้งหมดกับจำนวนประชากรที่อาศัยอยู่



ผลลัพธ์ที่ได้มีความสอดคล้องกับค่า correlation จาก heat-map ข้างต้น นั่นคือจำนวนห้องพักและจำนวนประชากรที่พักอาศัยมีความสัมพันธ์แปรผันตรง

- Scatterplot ระหว่างค่าเฉลี่ยของราคาบ้านและรายได้เฉลี่ยของครัวเรือน



จากผลลัพธ์ถึงแม้ว่ารายได้ของผู้พักอาศัยและมูลค่าของบ้านจะมีค่า correlation อยู่ที่ 0.62 จาก heat-map ข้างต้น แต่ทั้งสอง field ก็ไม่ได้มีความสัมพันธ์ที่ชัดเจนมากนัก

Data Modeling

❖ Linear Regression Method

- ทำการแทนที่ค่าใน column ocean_proximity เดิมให้เป็นตัวเลข 1 - 4

```
df_prep = df

ocean_prox_mapping = {'<1H OCEAN': 1, 'INLAND': 2, 'NEAR OCEAN': 3, 'NEAR BAY': 4}
df_prep['ocean_proximity'] = df['ocean_proximity'].replace(ocean_prox_mapping)
df_prep.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	4
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	4
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	4
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	4
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	4

- ทำการแบ่งข้อมูลทั้งหมดเป็น 2 ส่วนคือ df_train และ df_test จากนั้นเตรียมข้อมูลที่เหลือเพื่อให้เหมาะสมกับการนำไปทำ linear regression และทำการ train ข้อมูลโดยใช้ฟังก์ชัน LinearRegression

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

df_train, df_test = train_test_split(df_prep, test_size=0.3)

scaler = StandardScaler()

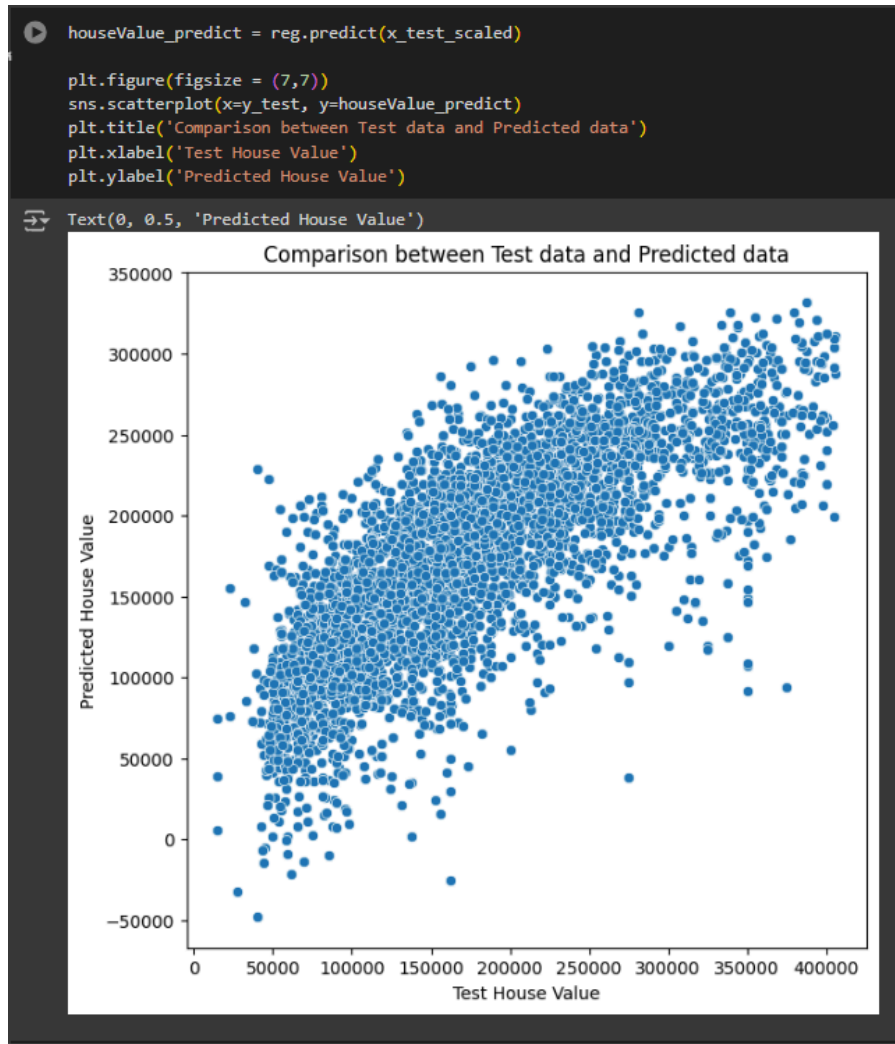
x_train = df_train.drop(columns='median_house_value')
y_train = df_train['median_house_value']
x_train_scaled = scaler.fit_transform(x_train)

x_test = df_test.drop(columns='median_house_value')
y_test = df_test['median_house_value']
x_test_scaled = scaler.fit_transform(x_test)

reg = LinearRegression()
reg.fit(x_train_scaled, y_train)
```

LinearRegression
LinearRegression()

- ทำการ predict house value ด้วย `x_test_scaled` จากนั้นนำค่าที่ predict ได้มาเทียบกับค่า house value จริง และแสดงการเปรียบเทียบค่าทั้ง 2 บน scatter diagram



- ทำการหาค่า R squared ซึ่งจะแสดงว่า model ที่เราทำการ train มานั้นดีมากน้อยเพียงใดโดยจะเปรียบเทียบค่าที่ predict กับ ค่าจริงแสดงผลเป็นเปอร์เซ็นต์ โดย model ของเราให้ผลเป็น 0.6 แสดงให้เห็นว่า model เรานั้นสามารถอธิบายการผันแปรของค่าตัวแปรตอบสนองที่กระจายตัวรอบค่าเฉลี่ยได้ปานกลาง และยังคำนวณค่า RMSE คือค่าที่แสดงถึง Error ของค่าที่ predict เทียบกับ ค่าจริงซึ่งมีค่าเป็น 52097

```
rmse = np.sqrt(mean_squared_error(y_test,houseValue_predict))
r2 = r2_score(y_test, houseValue_predict)
print(f"R squared score : {r2}\nRMSE : {rmse}")
```

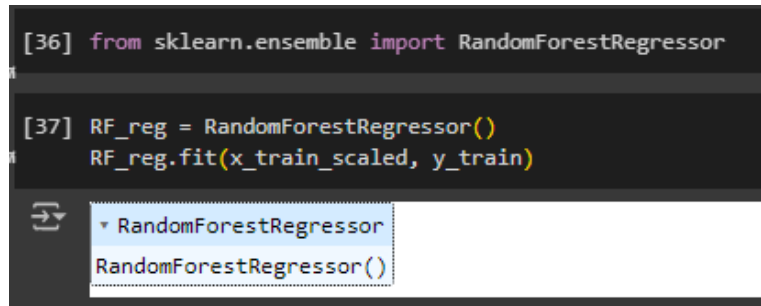
R squared score : 0.5972039031286094
RMSE : 52097.11923293729

❖ Random Forest Method

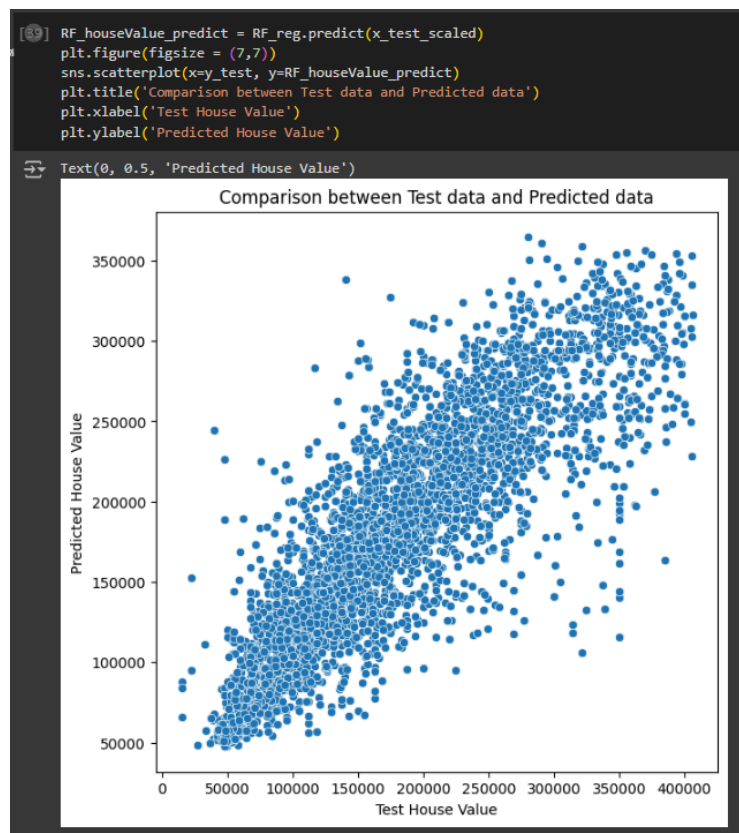
- ทำการ train model ด้วยวิธี Random Forest จากข้อมูล `x_train_scaled` และ `y_train` เดิมที่ได้ทำการ split มาตอนแรก ด้วยฟังก์ชัน `RandomForestRegressor`

```
[36] from sklearn.ensemble import RandomForestRegressor

[37] RF_reg = RandomForestRegressor()
      RF_reg.fit(x_train_scaled, y_train)
```

A screenshot of a Jupyter Notebook cell. The first line of code imports RandomForestRegressor from sklearn.ensemble. The second line creates an instance of RandomForestRegressor named RF_reg. The third line calls the fit method on RF_reg with x_train_scaled and y_train as arguments. Below the code, the Jupyter interface shows the object type as RandomForestRegressor and the method RandomForestRegressor().

- ทำการ predict house value ด้วย `x_test_scaled` จากนั้นนำค่าที่ predict ได้มาเทียบกับค่า house value จริง และแสดงการเปรียบเทียบค่าทั้ง 2 บน scatter diagram



- ทำการหาค่า R squared ซึ่งจะแสดงว่า model ที่เราทำการ train มานั้นดีมากน้อยเพียงใดโดยจะเปรียบเทียบค่าที่ predict กับ ค่าจริงแสดงผลเป็นเปอร์เซ็นต์ โดย model ของเราให้ผลเป็น 0.75 แสดงให้เห็นว่า model เรานั้นสามารถอธิบายการผันแปรของค่าตัวแปรตอบสนองที่กระจายตัวรอบค่าเฉลี่ยได้ดี และยังคงคำนวณค่า RMSE คือค่าที่แสดงถึง Error ของค่าที่ predict เทียบกับ ค่าจริงซึ่งมีค่าเป็น 40727

```
[ ] rmse = np.sqrt(mean_squared_error(y_test,RF_houseValue_predict))
r2 = r2_score(y_test, RF_houseValue_predict)
print(f"R squared score : {r2}\nRMSE : {rmse}")

R squared score : 0.7538300741104115
RMSE : 40727.577602059224
```

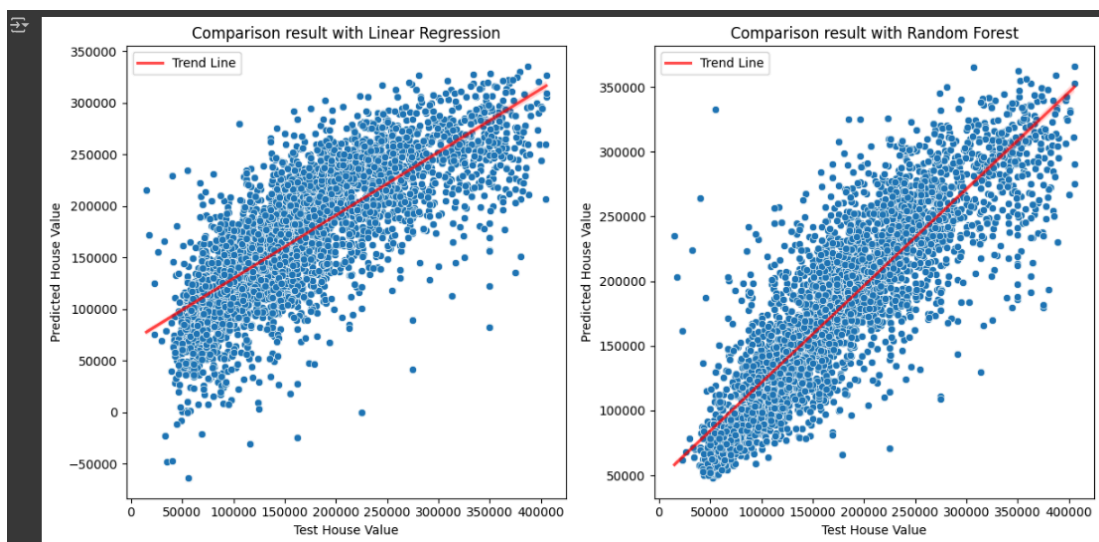
- ทำการเปรียบเทียบผลของการทำ model ด้วยวิธี linear regression และ random forest ซึ่งจากผลลัพธ์ของ R squared และ RMSE แล้วจะเห็นได้ว่าการทำ modeling แบบ random forest จะให้ผลลัพธ์ที่ดีกว่า และ error น้อยกว่าแบบ linear regression

```
[4] fig, axes = plt.subplots(1, 2, figsize=(12, 6)) # 1 row, 2 columns

# First subplot (scatter plot with trend line for houseValue_predict)
sns.scatterplot(x=y_test, y=houseValue_predict, ax=axes[0])
sns.regplot(x=y_test, y=houseValue_predict, scatter=False, color='red', label='Trend Line', line_kws={'alpha': 0.7}, ax=axes[0])
axes[0].set_title('Comparison result with Linear Regression')
axes[0].set_xlabel('Test House Value')
axes[0].set_ylabel('Predicted House Value')
axes[0].legend()

# Second subplot (scatter plot with trend line for RF_houseValue_predict)
sns.scatterplot(x=y_test, y=RF_houseValue_predict, ax=axes[1])
sns.regplot(x=y_test, y=RF_houseValue_predict, scatter=False, color='red', label='Trend Line', line_kws={'alpha': 0.7}, ax=axes[1])
axes[1].set_title('Comparison result with Random Forest')
axes[1].set_xlabel('Test House Value')
axes[1].set_ylabel('Predicted House Value')
axes[1].legend()

# Adjust layout and display the figure
plt.tight_layout()
plt.show()
```



Conclusion

ในการทำโปรเจกต์นี้เราได้เริ่มต้นจากการหาข้อมูลที่เหมาะสมและได้เป็นข้อมูลเกี่ยวกับราคาของบ้านในรัฐแคลิฟอร์เนีย ประเทศสหรัฐอเมริกา หลังจากนั้นก็ทำการศึกษาข้อมูลนี้และทำการทำความสะอาดข้อมูลแล้วจึงนำข้อมูลไปวิเคราะห์ต่อโดยทำ EDA และ Visualization เพื่อดูความสัมพันธ์ของข้อมูลและสิ่งที่น่าสนใจในข้อมูลชุดนี้ เมื่อทราบหัวข้อที่น่าสนใจจากการทำ EDA แล้วจึงทำการเตรียมข้อมูลเพื่อทำ model เพื่อทำนายมูลค่าของบ้าน โดยในโปรเจกต์นี้จะทำ model ด้วย 2 วิธีนั่นคือ Linear Regression และ Random Forest โดยใช้ข้อมูลชุดเดียวกันในการทำ model และ ทดสอบ model ซึ่งผลลัพธ์ที่ออกมาจะเห็นได้ว่าจากข้อมูลชุดนี้การทำ model แบบ Random Forest จะให้ค่า R squared ที่มากกว่าซึ่งแสดงให้เห็นถึงประสิทธิภาพในการทำนายของ model ที่สร้างด้วยวิธีนี้ว่าดีกว่า model ที่สร้างด้วยวิธี Linear Regression อีกทั้ง Random Forest ยังให้ค่าการผิดพลาดเมื่อเทียบการทำนายกับค่าจริง ที่น้อยกว่าการสร้าง model แบบ Linear Regression อีกด้วย

ซึ่งหากจะพัฒนา model ให้มีประสิทธิภาพในการทายเพิ่มขึ้นนั้นสามารถทำได้หลายวิธีไม่ว่าจะเป็นการพิจารณา method อื่น ๆ ในการสร้าง model ทำนาย หรือทำการเพิ่มคุณภาพของข้อมูลซึ่งอาจต้องพิจารณาข้อมูลที่ได้ในแต่ละ field เพิ่มเติม หรืออาจต้องการข้อมูลสำหรับการ train ที่มากขึ้น