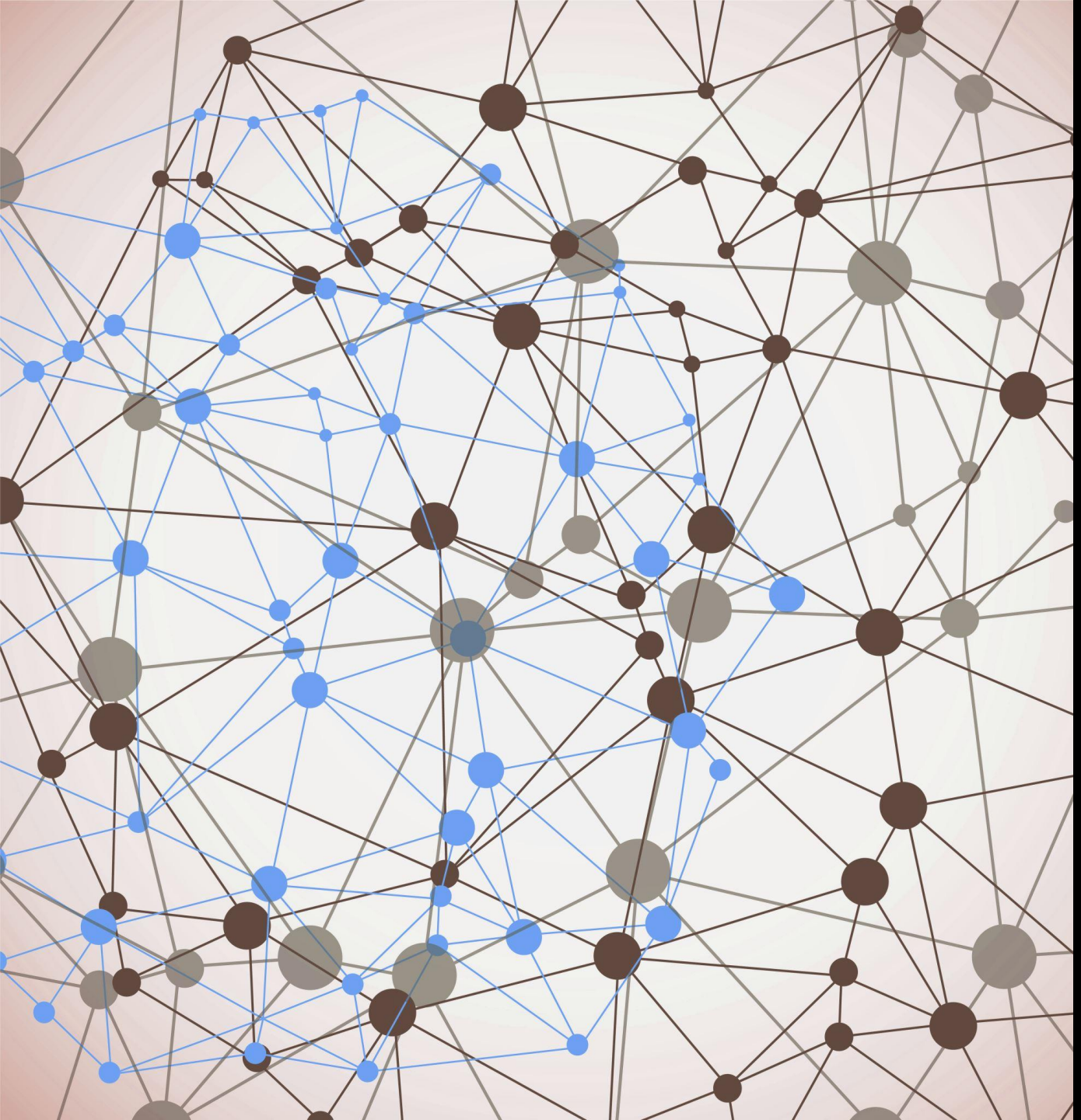


פרויקטון גמר- למידת מכונה

חיזוי מחיר נדלן בארהב

A	B	C	D	E	F	G	H	I	J	K	L	M
status	price	bed	bath	acre_lot	full_addres	street	city	state	zip_code	house_size	sold_date	
for_sale	105000	3	2	0.12	Sector Yah	Sector Yah	Adjuntas	Puerto Ric	601	920		
for_sale	80000	4	2	0.08	Km 78 9 Ca	Km 78 9 Ca	Adjuntas	Puerto Ric	601	1527		
for_sale	67000	2	1	0.15	556G 556-	556G 556-	Juana Diaz	Puerto Ric	795	748		
for_sale	145000	4	2	0.1	R5 Comun	R5 Comun	Ponce	Puerto Ric	731	1800		
for_sale	65000	6	2	0.05	14 Navarro	14 Navarro	Mayaguez	Puerto Ric	680			

הנתונים שקיבלנו

הצגת הנתונים בצורה מסודרת

בתחילה, ייבאנו את המידע שקיבלנו אל תוך הקוד שלנו

```
# Load CSV into DataFrame
file_path = 'realtor-dataset-100k.csv'  #path to the csv file
df = pd.read_csv(file_path)
```

לאחר מכן, כדי להתחיל ב "CLEANSING" של הנתונים הדפסנו את כמות הפיצ'רים שלנו ואת כמות ה "NONE NULL" כדי לברר כמה מידע חסר לנו בחלק מהפיצ'ר

```
 0   status        100000 non-null  object
 1   price         100000 non-null  int64
 2   bed           75050 non-null   float64
 3   bath          75112 non-null   float64
 4   acre_lot      85987 non-null   float64
 5   full_address  100000 non-null  object
 6   street        99916 non-null   object
 7   city          99948 non-null   object
 8   state         100000 non-null  object
 9   zip_code      99805 non-null   float64
 10  house_size    75082 non-null   float64
 11  sold_date     28745 non-null   object
dtypes: float64(5), int64(1), object(6)
memory usage: 9.2+ MB
```

מילוי הנתונים החסרים

לאחר שראינו שישנם נתונים שחסרים בשורות בחנו היטב את האפשרויות הניצבות בפנינו, השלמת הנתונים החסרים דרך השגתם מנתונים אחרים שקיימים או השמטתם ממאגר הנתונים שלנו,
בתחילה ראינו שיש בכל השורות FULL ADDRESS אך חסרים רחובות,ערים, ומיקוד של כ-100 שורות,

ניסינו לקבל את הנתונים החסרים מתוך הכתובת המלאה אך לאחר שביצענו זאת באמצעות פייתון ראינו שגם בכתובת המלאה פרטים אלו אינם היו קיימים ולכן בחרנו בסופו של דבר להשמיט את ה100~ שורות האלו (מתוך K100 שורות לא תהיה השפעה גדולה מדי על הסטטיסטיקה שלנו)

מילוי הנתונים החסרים המשך:

לעומת זאת, נתונים כגון כמות המיטות, גודל השטח, גודל הבית השלמנו באמצעות הנתונים הקיימים.

כיצד עשינו זאת?
השתמשנו בלוגיקה פשוטה, השתמשנו בחציון,

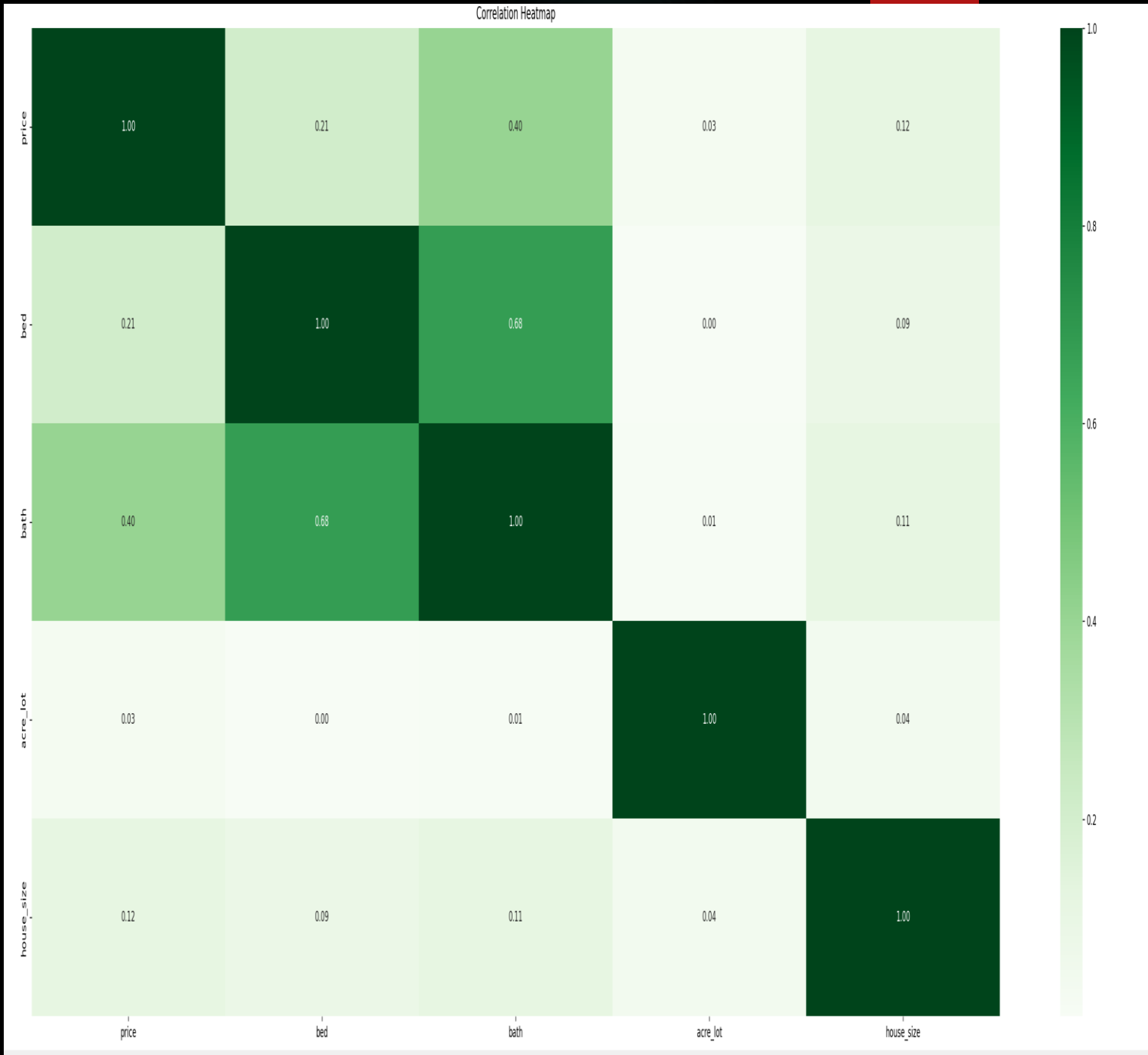
קודם כל במידה ונתונים אלו היו קיימים באותה עיר אז לקחנו חציון של אותה עיר, אם לא היו נתונים כלל באותה עיר עברנו לחציון של אותה מדינה רלוונטית במידה וגם זה לא קיים לקחנו חציון גלובלי (של כל אותה עמודה רלוונטית,קוד בעמוד הבא)

קוד להוספת הנתונים

```
######filling missing acre lot#########################################
# Calculate global, state, and city medians
global_median_acre_lot = df['acre_lot'].median()
state_medians_acre_lot = df.groupby('state')['acre_lot'].median().to_dict()
city_medians_acre_lot = df.groupby('city')['acre_lot'].median().to_dict()

acre_lot_medians = df['acre_lot'].mean()
print(f"the average acre_lot before is:{acre_lot_medians}")
# Custom function for imputation logic
def impute_acre_lot(row):  1 usage
    if not np.isnan(row['acre_lot']):  # If bath value is already present, return it
        return row['acre_lot']

    # Step 1: Check city-specific median
    if row['city'] in city_medians_acre_lot and not np.isnan(city_medians_acre_lot[row['city']]):
        return city_medians_acre_lot[row['city']]

    # Step 2: Check state-specific median
    if row['state'] in state_medians_acre_lot and not np.isnan(state_medians_acre_lot[row['state']]):
        return state_medians_acre_lot[row['state']]

    # Step 3: Fallback to global median
    return global_median_acre_lot
df['acre_lot'] = df.apply(impute_acre_lot, axis=1)
```

קורולוציה

שלב הבא בנסיון הסקת מסקנות וניקיון הנתונים שקיבלנו היה קורולציה בין הפיצ'רים השונים ולראות את הקשר ביניהם לבין המחיר:

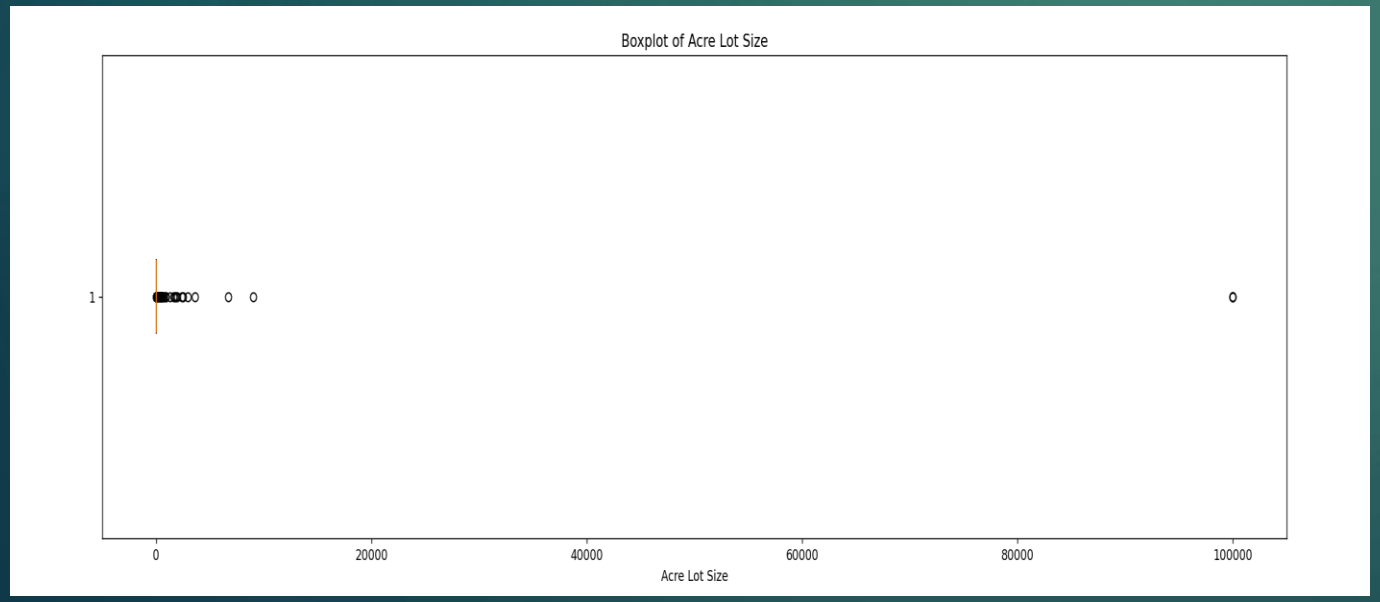


ה\אבחנה בעקבות מטריצת הקורולציה

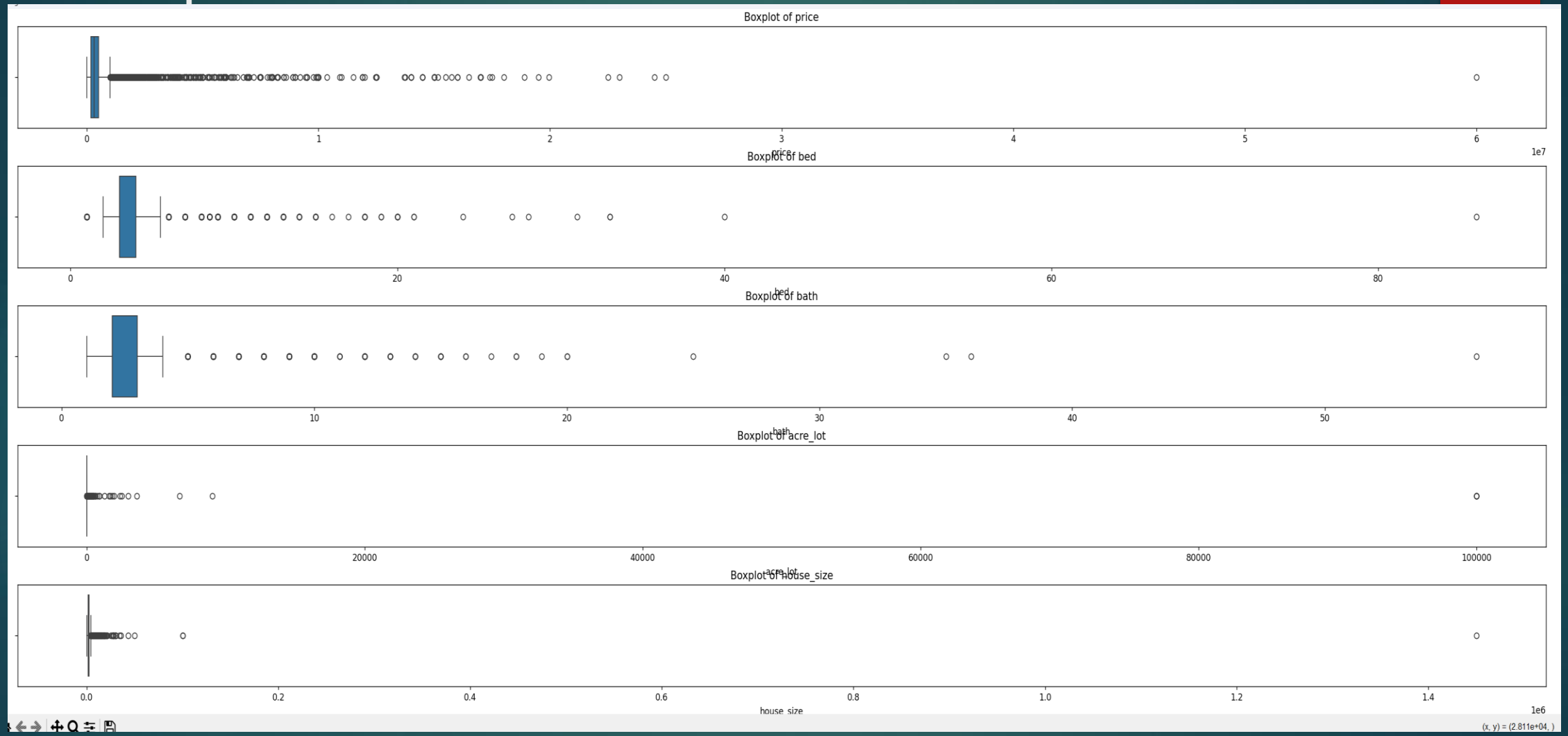
כפי שניתן היה להבחין בשקופית הקודמת הקשר בין המחיר של הבית לבין גודלו של הבית או של השטח היה ללא קורולציה כמעט מה שגרם לנו להסיק שעלינו "לנקות" את הנתונים שלנו טוב יותר ושישנן שורות שעדיין מוסיפות לנו "רעש" לסטטיסטיקה.

לכן, הדפסנו את הגודל הגבוה ביותר בעמודה של גודל המגרש להפתעתנו (או שלא)

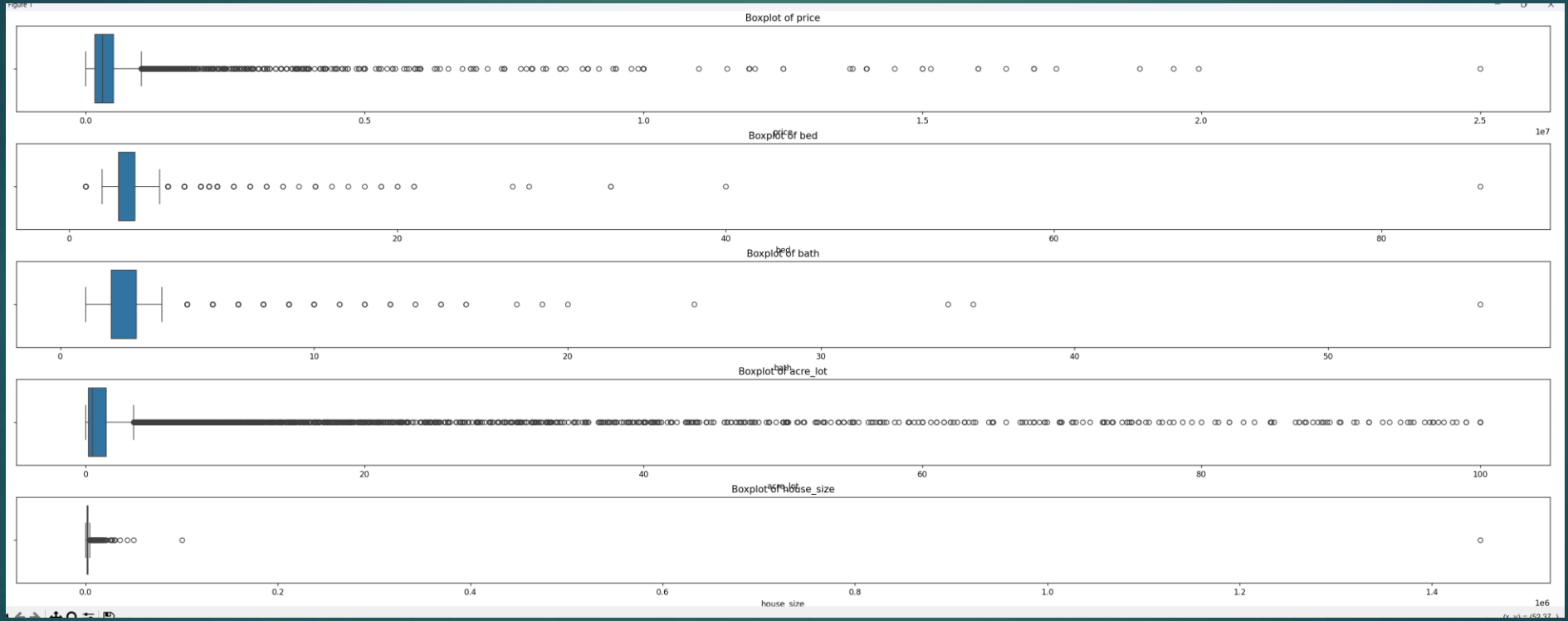
קיבלנו גדלים לא הגיונים (מאחר ומדובר ב ACRE)



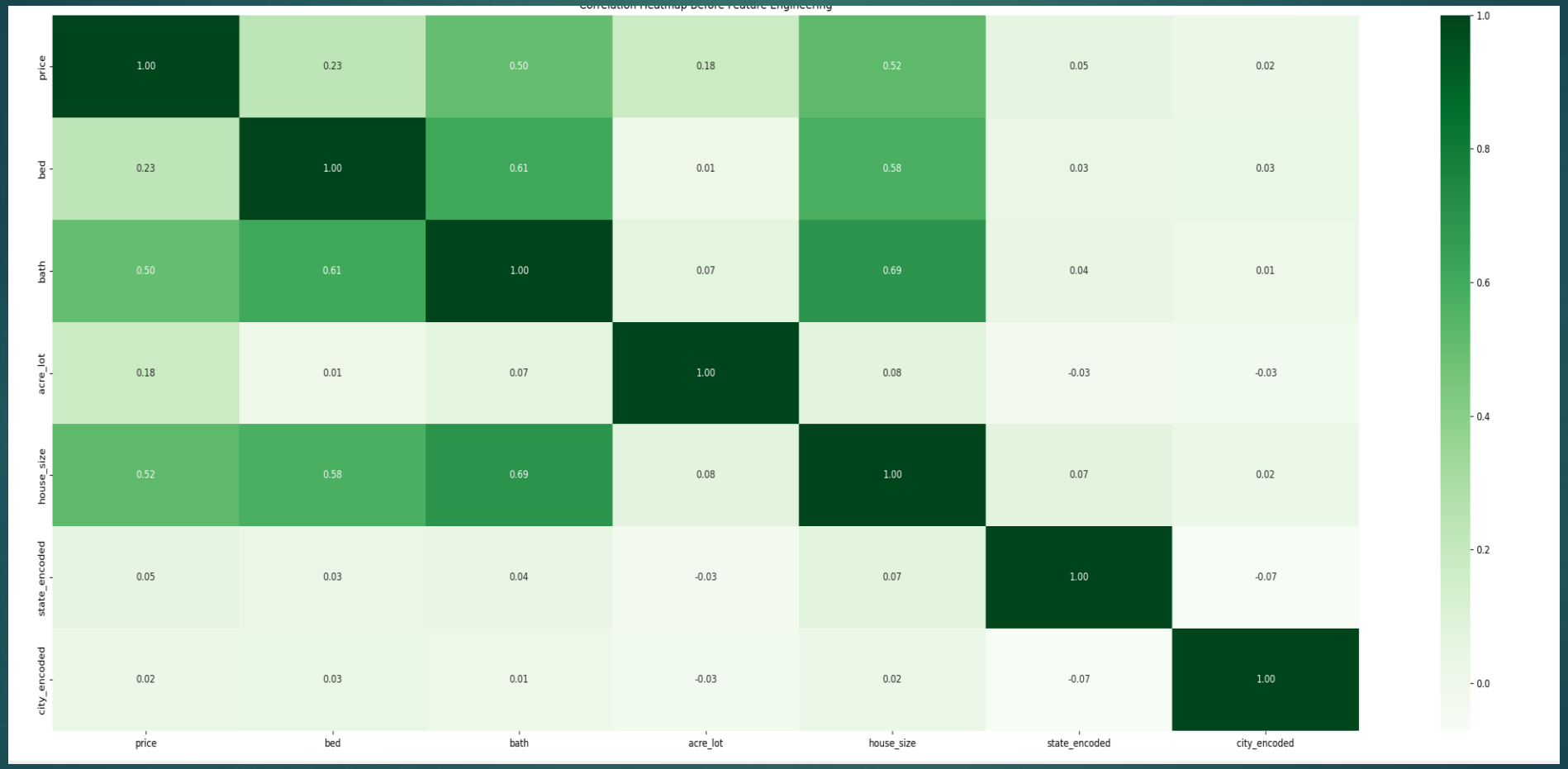
Box plot


Boxplot of price
0
1
2
3
4
5
6
price
1e7
Boxplot of bed
0
20
40
60
80
bed
Boxplot of bath
0
10
20
30
40
50
bath
Boxplot of acre_lot
0
20000
40000
60000
80000
100000
acre_lot
Boxplot of house_size
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
house_size
1e6
(x, y) = (2.811e+04,)

ולאחר שהורדנו את ה "WHISKERS"



מטריצת קורלציה אחרי הורדת הWHISKERS



חלוקה של הנתונים לשתי קבוצות(TEST/TRAINING)

```
# Split the data into training and test sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
```

אימון מודלים שונים והשוואת התוצאות

```
Training Linear Regression...
Linear Regression - MSE: 316883791751.12, R²: 0.33

Training Lasso Regression...
Lasso Regression - MSE: 316883846242.54, R²: 0.33

Training Ridge Regression...
Ridge Regression - MSE: 316883912966.51, R²: 0.33

Training Decision Tree Regressor...
Decision Tree - MSE: 334202634634.95, R²: 0.29

Training Random Forest Regressor...
Random Forest - MSE: 237174508174.58, R²: 0.50

Training XGBoost Regressor...
XGBoost Regressor - MSE: 220836974559.59, R²: 0.53
```

מסקנות מתוצאות המודל הנמוכות

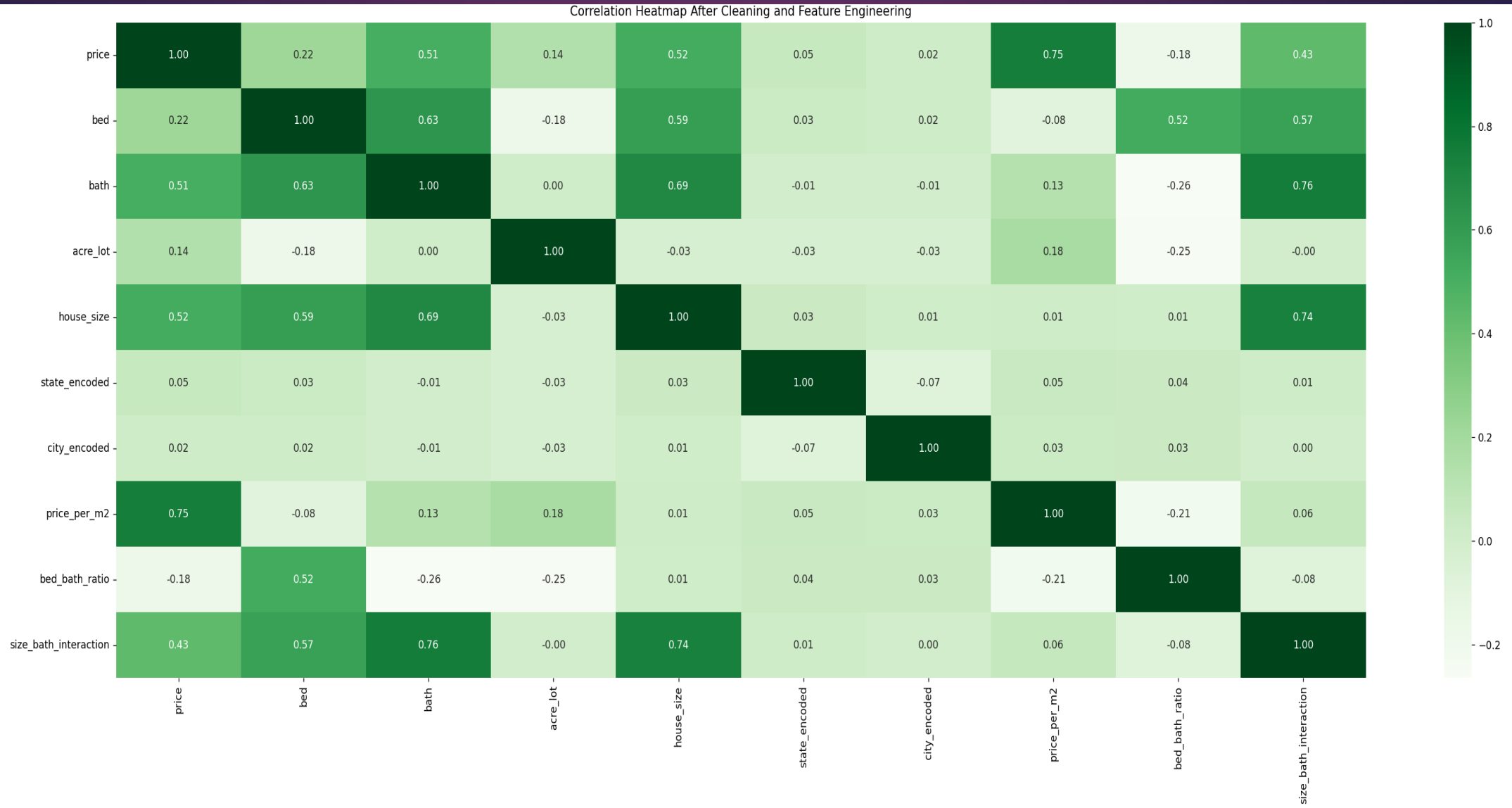
ישנה בעיה מהותית וקורולוציה נמוכה מדי בין הפיצ'רים השונים לבין מחיר הבית לכן עלינו להוסיף פיצ'רים נוספים מתוך הקיימים כדי לשפר את יכולת חיזוי המחיר של המודל.

Feature Engineering

אז מאחר והחלטנו שעלינו להוסיף פיצ'רים מתוך הפיצ'רים הקיימים הבנו שעלינו לבחור בפיצ'רים עם הקורולוציה הכי גבוהה בינם לבין המחיר.

```
# Step 9: Feature Engineering
print("\nFeature engineering: Creating new features...")
df['price_per_m2'] = df['price'] / (df['house_size'] + 1)  # Prevent division by zero
df['bed_bath_ratio'] = df['bed'] / (df['bath'] + 1)  # Prevent division by zero
df['size_bath_interaction'] = df['house_size'] * df['bath']
```

HEATMAP לאחר הוספת הפיצ'רים



תוצאות המודל לאחר הוספת הפיצ'רים

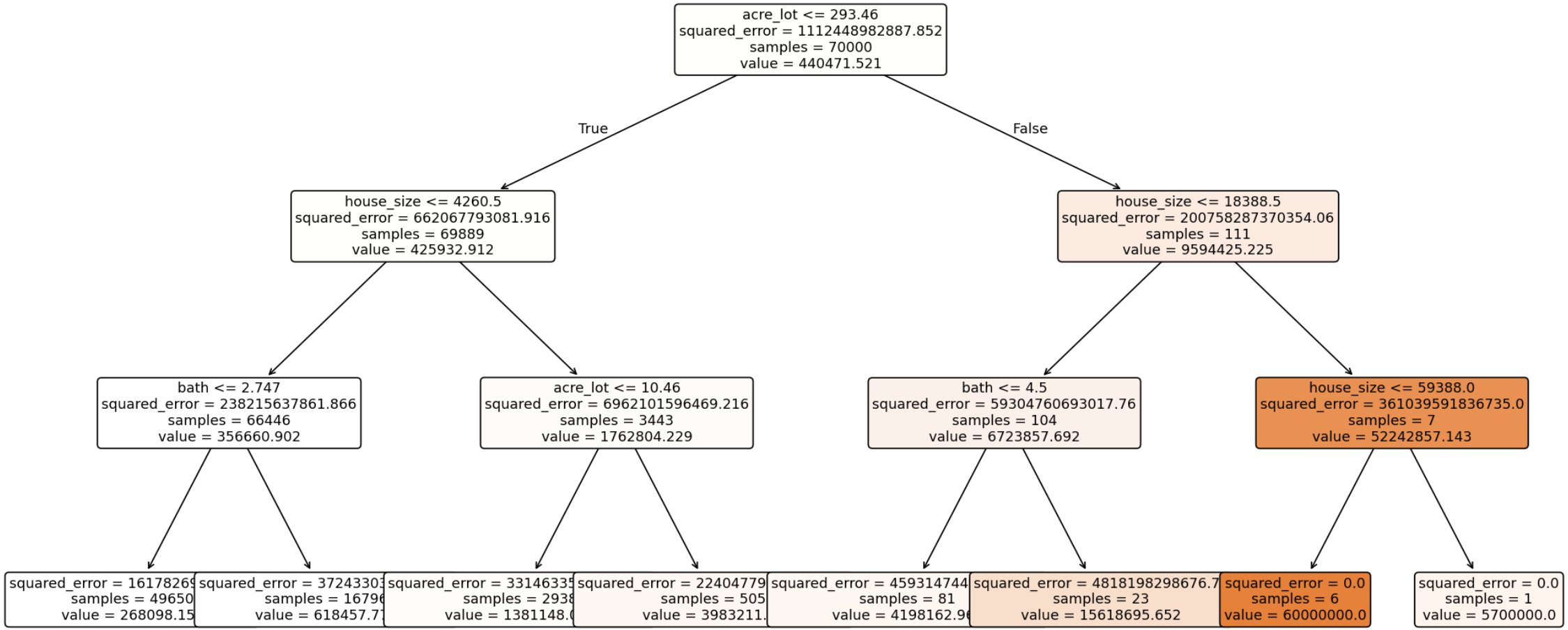
Model	Mean Squared Error (MSE)	R^2 Score
Linear Regression	1,069,330,087.91	0.77
Lasso Regression	1,069,333,100.89	0.77
Ridge Regression	1,069,362,011.35	0.77
Decision Tree	1,823,193,463.09	0.96
Random Forest	916,184,325.38	0.98
XGBoost	1,969,884,456.51	0.96

כפי שניתן לראות בתוצאות לאחר הנדסת הפיצ'רים החדשים
התוצאות של המודלים השונים השתפרו משמעותית המודלים הלינארים השתפרו מ=~0.33 ל R=0.77

מודלים מבוססי עץ החלטות RANDOM FOREST, DECISION TREE,XGBOOST
תפקדו טוב יותר כאשר RANDOM FOREST מוביל בביצועים עם תוצאה של 0.98

מדוע XGBOOST כ'כ טוב?
למרות שבביצועים על נתונים אלו ביצועיו לא עלו על אלו של RANDOM FOREST, XGBOOST מצטיין בעבודה עם מאגרי נתונים גדולים
ובהורדת ה OVERFITTING דרך טכניקות של רגולציה מה שגורם למודל להיות גמיש ולהתאים להרבה סוגים שונים של מאגרי נתונים.

Decision Tree Visualization



Price prediction Vs. actual price