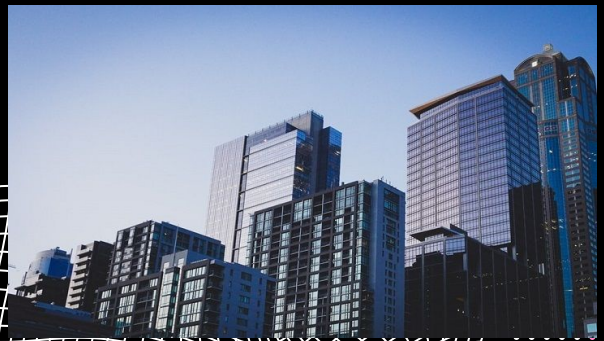
Real Estate Rent Price Prediction

Agustinus Angelo Christian Fernando - 21/473804/TK/52235
Aufa Nasywa Rahman - 21/475255/TK/52454
Ahmad Zaki Akmal - 21/480179/TK/52981



Latar Belakang

- Kami mengimplementasikan 2 tema yang disediakan di soal, yaitu tema *money* dan *investment*
- Orang yang ingin menyewa bangunan pasti mencari bangunan dengan harga sewa paling rendah, namun dengan keuntungan paling banyak.
- Tujuannya agar penyewa bisa menghemat uang yang mereka miliki dan mereka bisa menikmati fasilitas yang sepadan dengan uang yang mereka keluarkan.
- Hal tersebut merupakan implementasi dari tema *money*.
- Investor tentunya akan membeli bangunan yang pasaran harga sewanya masih tinggi dan memiliki prospek yang bagus untuk jangka panjang.
- Hal tersebut merupakan implementasi dari tema *investment*.

Latar Belakang

- Jika dilihat, ada kesamaan pada kebutuhan penyewa dan investor.
- Kesamaannya adalah keduanya membutuhkan prediksi harga sewa dari jenis-jenis bangunan yang mereka inginkan.
- Oleh karena itu, kami ingin memprediksi harga sewa bangunan yang ada di sebuah kota besar, yaitu New York, lebih tepatnya di daerah Queens.
- Prediksi harga sewa bangunan ini diharapkan dapat membantu para penyewa dan investor untuk mencari bangunan yang mereka inginkan.

Deskripsi Dataset

- Sumber dataset: https://github.com/Codecademy/datasets/tree/master/streeteasy
- Dataset ini berisi tentang data bangunan di wilayah Queens, New York.
- Dataset ini diambil oleh StreetEasy yang merupakan salah satu marketplace Real Estate terbesar di New York.
- Bangunan yang ada di sini beragam, mulai dari yang tipe studio sampai bangunan besar bertingkat.
- Ada 448 data bangunan di dataset ini
- Apa saja feature yang ada? Mari cek slide berikutnya

Deskripsi setiap kolom dataset

1. rental_id: Id setiap penyewaan.
2. rent: Harga sewa bangunan dalam mata uang Dollar Amerika.
3. bedrooms: Jumlah kamar tidur di bangunan tersebut.
4. bathrooms: Jumlah kamar mandi di bangunan tersebut.
5. size_sqft: Luas bangunan dalam satuan square feet.
6. min_to_subway: Waktu yang dibutuhkan untuk ke subway (dalam satuan menit)
7. floor: lantai yang disewakan
8. building_age_yrs: Umur bangunan tersebut
9. no_fee: Apakah ada fee untuk broker? (0 jika ada, 1 jika tidak ada)

Deskripsi setiap kolom dataset

9. has_roofdeck: Apakah ada roof deck? (0 jika tidak ada, 1 jika ada)
10. has_washer_dryer: Apakah ada washer atau dryer? (0 jika tidak ada, 1 jika ada)
11. has_doorman: Apakah ada doorman? (0 jika tidak ada, 1 jika ada)
12. has_elevator: Apakah ada elevator? (0 jika tidak ada, 1 jika ada)
13. has_dishwasher: Apakah ada pencuci piring? (0 jika tidak ada, 1 jika ada)
14. has_patio: Apakah ada teras? (0 jika tidak ada, 1 jika ada)
15. has_gym: Apakah ada gym? (0 jika tidak ada, 1 jika ada)
16. neighborhood: Nama daerah tempat bangunan berada
17. borough: Nama wilayah tempat bangunan berada

Tujuan

- Kami ingin memprediksi harga sewa bangunan di daerah Queens, New York dengan metode *Linear Regression*.
- Kami juga ingin melakukan *Exploratory Data Analysis* (EDA) agar bisa mendapatkan berbagai *insight* yang bermanfaat dari data ini
- Kami melakukan visualisasi beberapa feature penting dari dataset ini agar *insight* dari data semakin jelas.
- Kami akan menampilkan evaluasi metriks untuk mengukur seberapa akurat model yang kita buat.
- Evaluasi yang akan kami gunakan adalah *R2 Score, Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE), dan *Mean Absolute Percentage Error* (MAPE).

Metode

- Metode yang kami gunakan adalah *Linear Regression*.
- **Alasan:**
- 1. Output dari hasil prediksi berupa harga sewa bangunan. Kita tahu bahwa harga sewa bangunan berupa *real number*. Maka, *linear regression* merupakan salah satu metode yang tepat.
- 2. Harga sewa memiliki relasi linear dengan feature-featurenya sehingga cocok menggunakan model *linear regression.*
- 3. Regresi linear memiliki kemampuan yang baik untuk mendeteksi outlier.
- 4. Akurat untuk data yang sudah dinormalisasi. Kami sudah melakukan normalisasi pada feature yang belum terdistribusi normal. Oleh karena itu, *linear regression* cocok digunakan untuk dataset ini.

Data Cleaning

1. Tidak ada value yang null
2. Mengubah feature bedrooms dan floor dari yang awalnya float menjadi integer karena jumlah kamar tidur dan jumlah lantai tidak mungkin desimal.

```
# Mengubah kolom bedrooms dan floor menjadi tipe data interger
konversi = ['bedrooms', 'floor']

for col in konversi:
  df[col] = df[col].astype(int)
```

Data Cleaning

3. Drop feature yang tidak diperlukan, yaitu borough dan rental_id.

   borough tidak diperlukan karena isinya sama semua, yaitu ‘Queens’.

   rental_id tidak diperlukan karena prediksi tidak memerlukan id.

```
df.drop(columns='borough', inplace=True)
df.drop(columns='rental_id', inplace=True)
```

4. Melakukan normalisasi pada feature building_age_yrs agar semakin optimal ketika menggunakan metode *linear regression*.

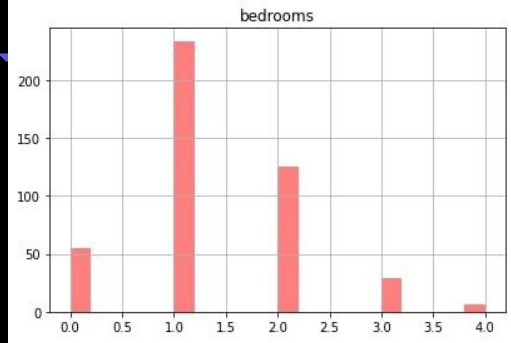
```
df['building_age_yrs'] = np.log1p(df['building_age_yrs'])
```

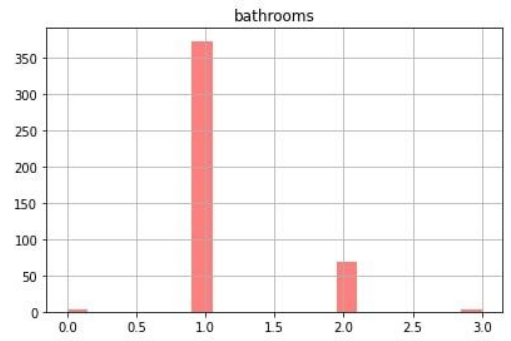
Data Cleaning

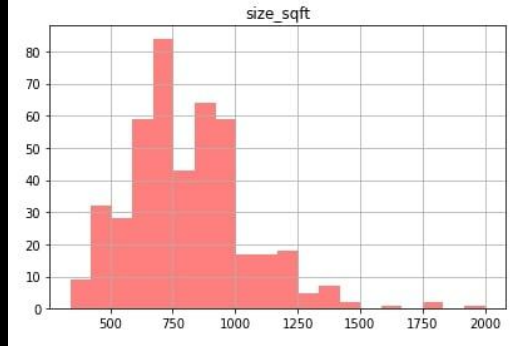
5. Melakukan encoding pada kolom non-numerik agar bisa dilakukan modelling dengan *linear regression*. Encoding dilakukan dengan menggunakan pd.get_dummies.

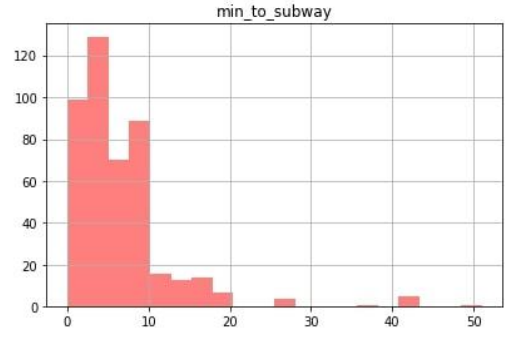
```
df = pd.get_dummies(df, columns=['neighborhood'], drop_first=True)
```

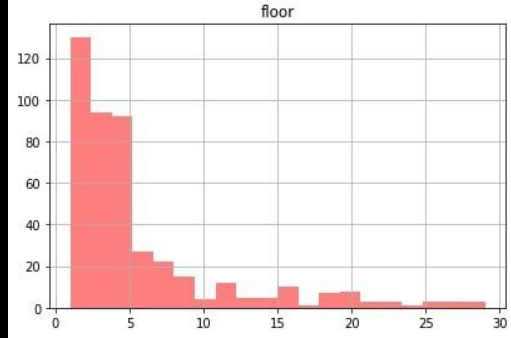
EDA - Visualisasi Histogram Setiap Feature Numerik

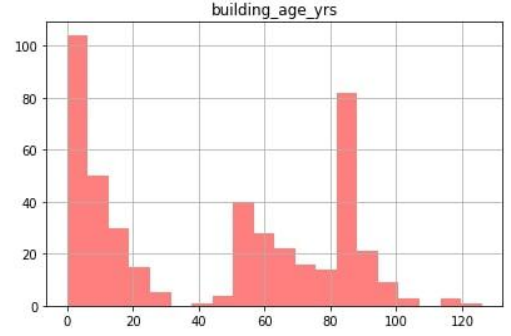




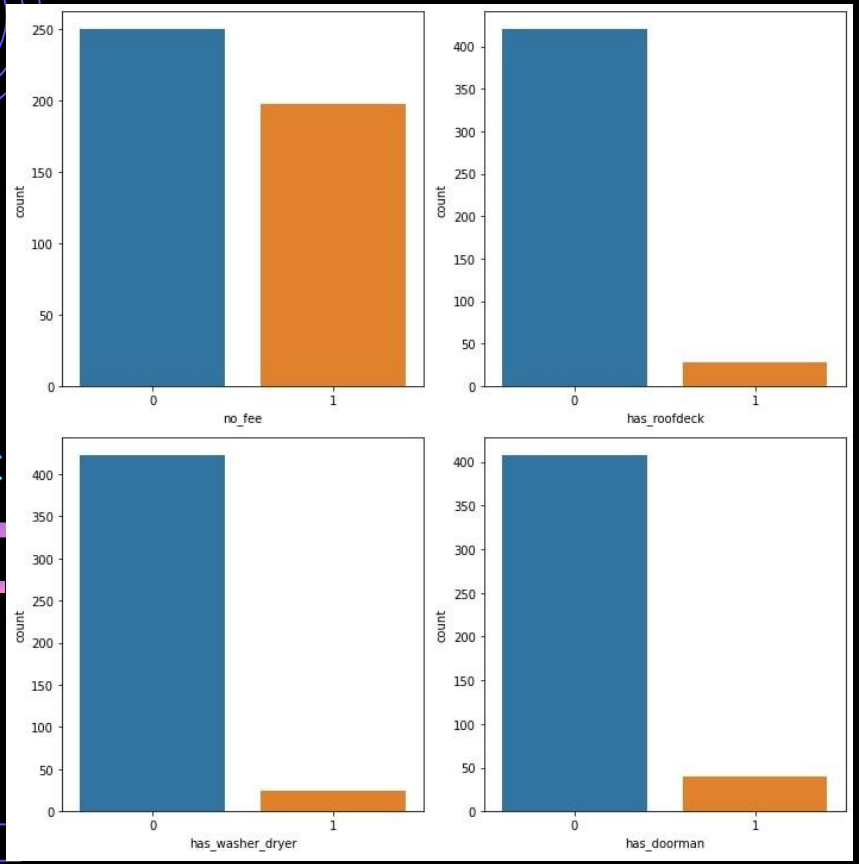


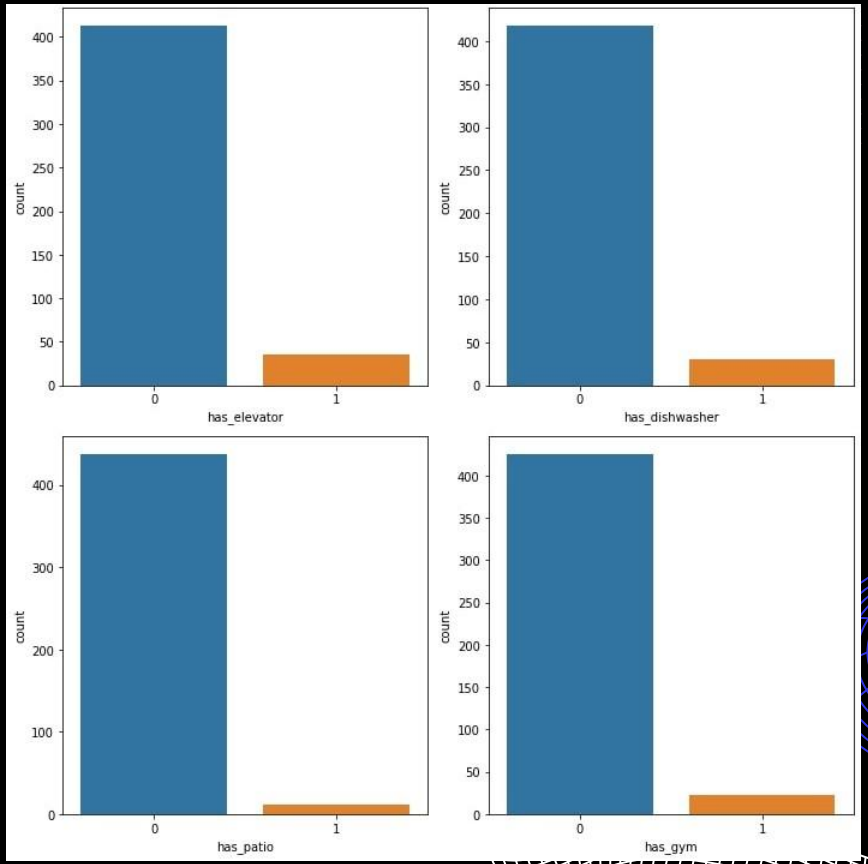






EDA - Visualisasi Count Plot Setiap Feature Kategorikal





EDA - Visualisasi Top 5 Wilayah Dengan Harga Sewa Termahal

Persiapan Modeling

- Membagi dataframe menjadi X dan y
- X adalah feature-feature yang digunakan. Artinya X memuat semua kolom di dataframe kecuali kolom rent.
- y adalah kolom yang akan kita prediksi, yaitu kolom rent (harga sewa).

```
X = df.drop(['rent'], axis = 1)
y = df['rent']

print(X.shape)
print(y.shape)
```

```
(448, 37)
(448,)
```

Standarisasi

- Standarisasi dilakukan dengan *Z-score* terhadap X
- Tujuannya untuk menghindari efek samping jika ada data yang esktrim sehingga akurasi dari *linear regression* bisa lebih akurat.

$$Z = \frac{x - \mu}{\sigma}$$

```python
# Fungsi untuk melakukan standarisasi
# Penjelasan parameter:
# df: dataframe yang akan distandarisasi
def standarisasi(df):
    return (df - df.mean()) / df.std()

# Melakukan standarisasi pada setiap feature di X
X = standarisasi(X)
```

Splitting Data

- Data dibagi menjadi X_train, X_test, y_train, dan y_test
- Persentasenya adalah:

  Data tes sebesar 20%

  Data train sebesar 80%

```
# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=92)
X_train.shape, X_test.shape
```

```
((358, 37), (90, 37))
```

Model Linear Regression

- Langkah pertama adalah mencari costnya terlebih dahulu
- *Cost function* ini digunakan untuk mengukur seberapa baik model belajar dan menyesuaikan ketika dilatih menggunakan data-data baru.
- Untuk model yang berdasar pada regresi linear, *cost function* yang digunakan adalah *mean squared error* (MSE). Rumusnya adalah seperti yang ditampilkan di atas.

$$J(\beta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\beta(x^{(i)}) - y^{(i)})^2$$

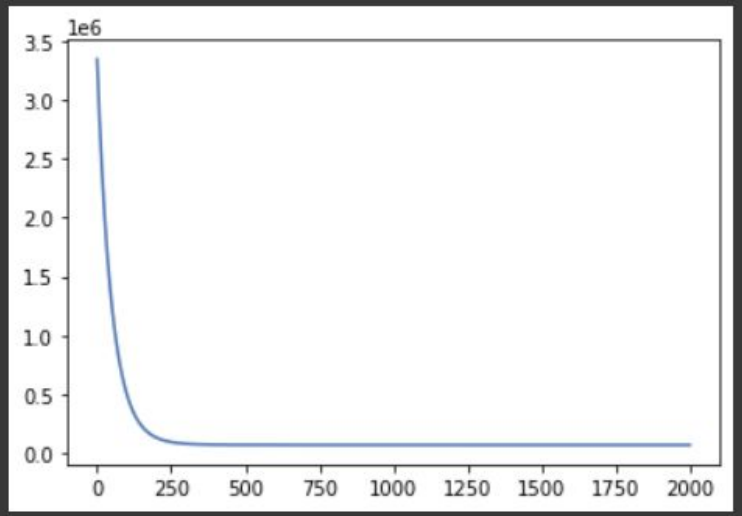
```
# Fungsi untuk mencari cost
# Penjelasan parameter:
# X: features yang digunakan
# y: nilai yang akan diprediksi
# w: weight
# b: bias
def cost_function(X, y, w, b):
    cost = np.sum((((X.dot(w) + b) - y) ** 2) / (2*len(y)))
    return cost
```

Model Linear Regression

- Memodelkan fungsi untuk gradient descent.
- Tujuannya untuk meminimalisir error pada model *linear regression*.
- Gradient descent ini akan melakukan iterasi pada setiap parameter agar bisa menemukan nilai parameter terbaik sehingga bisa meminimalisir error atau loss pada model.
- Dalam kasus regresi linear, gradient descent berguna dalam menyesuaikan nilai-nilai parameter yang biasanya disimbolkan dengan ß.
- Iterasi ini akan dilakukan hingga konvergen ke suatu nilai *cost function* yang minimum.

Model Linear Regression

```
# Memvisualisasikan costnya
plt.plot(costs)
plt.show()
```



- Tampak bahwa costnya berkurang dengan sangat cepat menuju 0 seiring bertambahnya iterasi.
- Hal tersebut membuktikan bahwa model yang kami buat mampu belajar dengan cepat ketika dilatih dengan data-data yang ada.

Model Linear Regression

$$z = X \cdot w + b$$

$$z = \begin{bmatrix} x11 & \cdots & xn1 \\ \vdots & \ddots & \vdots \\ x1m & \cdots & xnm \end{bmatrix} \cdot \begin{bmatrix} ß1 \\ ß2 \\ \vdots \\ ßn \end{bmatrix} + ß0$$

m x n n x 1

```python
def gradient_descent_function(X, y, w, b, alpha=0.01, epochs=1000):
    m = len(y)
    costs = [0] * epochs

    for epoch in range(epochs):
        # Calculate the value -- Forward Propagation
        z = X.dot(w) + b

        # Calculate the losses
        loss = z - y

        # Calculate gradient descent
        weight_gradient = X.T.dot(loss) / m
        bias_gradient = np.sum(loss) / m

    return w, b, costs
```

- z = vektor berisi nilai prediksi y
- x = nilai features
- m = banyak data
- n = banyak *features*
- ß = koefisien parameter

$$z = \begin{bmatrix} x11.ß1 + \cdots + xn1.ßn + ß0 \\ \vdots \\ x1m.ß1 + \cdots + xnm.ßn + ß0 \end{bmatrix}$$

m x 1

Model Linear Regression

$$loss = z - y$$

$$loss_{\text{m x 1}} = \begin{bmatrix} z1 \\ z2 \\ \vdots \\ zm \end{bmatrix} - \begin{bmatrix} y1 \\ y2 \\ \vdots \\ ym \end{bmatrix}$$

z = vektor berisi nilai prediksi y
y = vektor berisi nilai asli y
loss = error, selisih antara prediksi dan kenyataan

```python
def gradient_descent_function(X, y, w, b, alpha=0.01, epochs=1000):
    m = len(y)
    costs = [0] * epochs

    for epoch in range(epochs):
        # Calculate the value -- Forward Propagation
        z = X.dot(w) + b

        # Calculate the losses
        loss = z - y

        # Calculate gradient descent
        weight_gradient = X.T.dot(loss) / m
        bias_gradient = np.sum(loss) / m

        # Update weights and bias
        w = w - alpha*weight_gradient
        b = b - alpha*bias_gradient

        # Store current lost
        cost = cost_function(X, y, w, b)
        costs[epoch] = cost

    return w, b, costs
```

Model Linear Regression

$$w\_gradient = \frac{\overset{n \times m}{X^T} \cdot \overset{m \times 1}{loss}}{m}$$

Hasil *dot product* dari X transpose dengan vektor loss akan menghasilkan vektor (nx1) yang tiap row nya berisi sum dari fiturnya dikalikan errornya, jika dibagi dengan m, akan didapat mean atau *gradient*-nya.

```python
def gradient_descent_function(X, y, w, b, alpha=0.01, epochs=1000):
    m = len(y)
    costs = [0] * epochs

    for epoch in range(epochs):
        # Calculate the value -- Forward Propagation
        z = X.dot(w) + b

        # Calculate the losses
        loss = z - y

        # Calculate gradient descent
        weight_gradient = X.T.dot(loss) / m
        bias_gradient = np.sum(loss) / m

        # Update weights and bias
        w = w - alpha*weight_gradient
        b = b - alpha*bias_gradient

        # Store current lost
        cost = cost_function(X, y, w, b)
        costs[epoch] = cost

    return w, b, costs
```

Model Linear Regression

$$b\_gradient = \frac{loss_1 + loss_2 + \cdots + loss_m}{m}$$

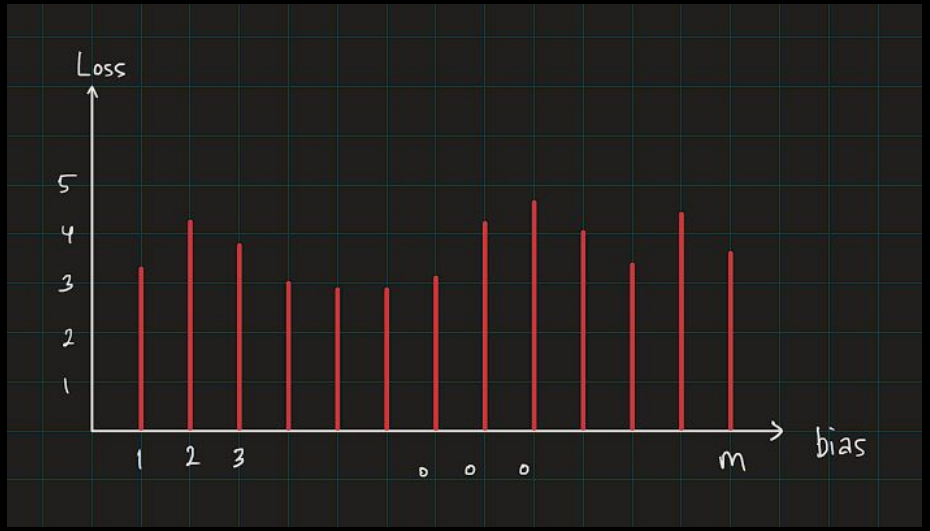
loss_m = elemen ke m dari vektor loss
m = banyak data

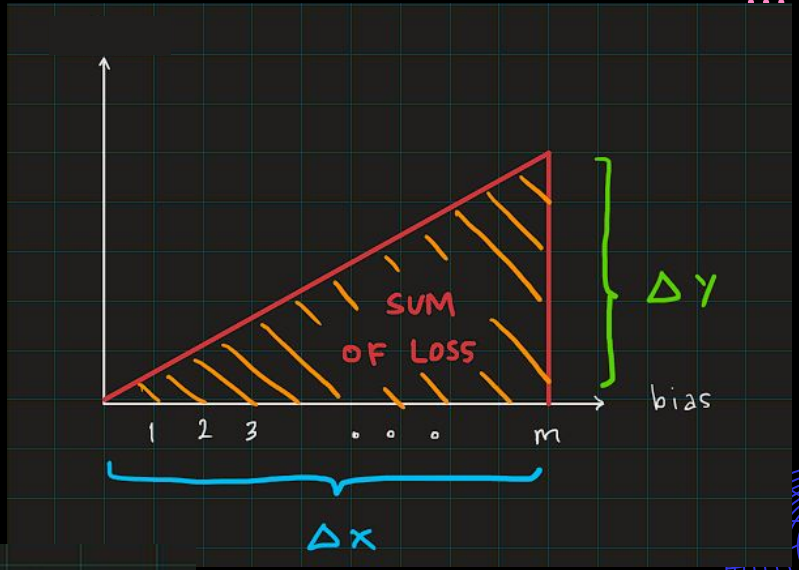
Dari rumusnya, terlihat bahwa *gradient* dari b ini dihitung dengan rumus rata-rata, yaitu jumlah semua data dibagi banyaknya.

```python
def gradient_descent_function(X, y, w, b, alpha=0.01, epochs=1000):
    m = len(y)
    costs = [0] * epochs

    for epoch in range(epochs):
        # Calculate the value -- Forward Propagation
        z = X.dot(w) + b

        # Calculate the losses
        loss = z - y

        # Calculate gradient descent
        weight_gradient = X.T.dot(loss) / m
        bias_gradient = np.sum(loss) / m

        # Update weights and bias
        w = w - alpha*weight_gradient
        b = b - alpha*bias_gradient

        # Store current lost
        cost = cost_function(X, y, w, b)
        costs[epoch] = cost

    return w, b, costs
```

Model Linear Regression

Mengapa di sini gradient dicari dengan rumus mean?





$$\text{gradient} = \frac{\Delta Y}{\Delta X} = \frac{\text{Sum of loss}}{m}$$

Model Linear Regression

$$\theta i = \theta i - \frac{\alpha \, \partial J(\theta)}{\partial \theta i}$$

$$\theta i = \theta i - \alpha(cost\ func.gradient)$$

θi = parameter yang dioptimalkan (ß0, ß1, ß2, ..., ßn)
α = *learning rate* dari model
J(θ) = *cost function*

- Nilai cost yang didapatkan disimpan

```python
def gradient_descent_function(X, y, w, b, alpha=0.01, epochs=1000):
    m = len(y)
    costs = [0] * epochs

    for epoch in range(epochs):
        # Calculate the value -- Forward Propagation
        z = X.dot(w) + b

        # Calculate the losses
        loss = z - y

        # Calculate gradient descent
        weight_gradient = X.T.dot(loss) / m
        bias_gradient = np.sum(loss) / m

        # Update weights and bias
        w = w - alpha*weight_gradient
        b = b - alpha*bias_gradient

        # Store current lost
        cost = cost_function(X, y, w, b)
        costs[epoch] = cost

    return w, b, costs
```

Evaluasi Model

- Ada 4 evaluasi yang kami lakukan:
- 1. R2 Score
- 2. *Mean Absolute Error* (MAE)
- 3. *Root Mean Square Error* (RMSE)
- 4. *Mean Absolute Percentage Error (*MAPE)
- Tujuannya agar kita tahu seberapa akurat model yang kita buat dan seberapa besar error yang terjadi.

R2 Score

- Tujuan dari R2 Score adalah agar kita bisa mengetahui seberapa akurat model yang kita buat.
- Berikut adalah rumus dan code untuk R2 Score:

```
# Fungsi untuk menghitung skor r2
# Tujuannya agar kita bisa tahu seberapa akurat model yang kita buat
# Penjelasan parameter:
# y_pred: value hasil prediksi
# y: value aktual
def r2score(y_pred, y):
    SSres = np.sum((y - y_pred) ** 2)
    SStot = np.sum((y - y.mean()) ** 2)

    r2 = 1 - (SSres / SStot)
    return r2


r2_test = r2score(y_pred_test, y_test)
r2_train = r2score(y_pred_train, y_train)
```

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Hasil R2 Score

```
print(f"Skor R2 untuk data test = {r2_test}")
print(f"Skor R2 untuk data train = {r2_train}")
```

```
Skor R2 untuk data test = 0.7929489350956764
Skor R2 untuk data train = 0.8128068113133003
```

- Tampak bahwa skor r2 untuk data test adalah 79% dan skor r2 untuk data train adalah 81%.
- Hal tersebut membuktikan bahwa model ini balanced.
- Tidak ada underfitting dan overfitting.
- Hal tersebut karena skor data test dan data train sangat mirip, serta skornya pun sudah cukup baik, yaitu ada di sekitar 80%.

Mean Absolute Error (MAE)

- Tujuan MAE adalah untuk memeriksa seberapa jauh perbedaan atau error antara harga sewa aktual dengan harga sewa hasil prediksi.
- Berikut adalah rumus dan kodenya:

```
# Fungsi untuk menghitung besar Mean Absolute Error
# Penjelasan parameter:
# y: value aktual
# y_pred: value hasil prediksi
def mean_absolute_error(y, y_pred):
    return np.mean(np.abs(y - y_pred))
```

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$

```
mae_test = mean_absolute_error(y_test, y_pred_test)
mae_train = mean_absolute_error(y_train, y_pred_train)
```

Hasil MAE

```
print(f"Mean absolute error untuk data tes = {mae_test}")
print(f"Mean absolute error untuk data train = {mae_train}")
```

```
Mean absolute error untuk data tes = 293.6927399587293
Mean absolute error untuk data train = 243.56688067501383
```

- Tampak bahwa rata-rata selisih antara harga sewa aktual dengan harga sewa hasil prediksi hanya sebesar 293 dollar untuk data test dan 243 dollar untuk data train.
- Hal tersebut menunjukkan bahwa error yang terjadi tidak terlalu besar dalam konteks harga sewa bangunan.
- Maka, terbukti model ini menghasilkan error yang minimal.

Root Mean Squared Error (RMSE)

- Tujuan RMSE adalah untuk menghitung akar kuadrat kesalahan antara harga sewa aktual dengan harga sewa hasil prediksi.
- Berikut adalah kode dan rumusnya:

```
# Fungsi untuk menghitung besar Mean Absolute Error
# Penjelasan parameter:
# y: value aktual
# y_pred: value hasil prediksi
def root_mean_squared_error(y, y_pred):
    return np.sqrt(np.mean((y - y_pred)**2))
```

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

```
rmse_test = root_mean_squared_error(y_test, y_pred_test)
rmse_train = root_mean_squared_error(y_train, y_pred_train)
```

Hasil RMSE

```
print(f"Root mean squared error untuk data tes = {rmse_test}")
print(f"Root mean squared error untuk data train = {rmse_train}")
```

```
Root mean squared error untuk data tes = 410.413637743121
Root mean squared error untuk data train = 361.87763156613886
```

- Tampak bahwa RMSE untuk data tes adalah 410 dollar dan RMSE untuk data train adalah 361 dollar.
- Hal tersebut membuktikan bahwa model mampu memprediksi harga sewa dengan baik karena besar errornya yang minim.

Mean Absolute Percentage Error (MAPE)

- Tujuan MAPE adalah untuk menghitung rata-rate persentase error antara harga sewa aktual dengan harga sewa hasil prediksi.
- Berikut adalah kode dan rumusnya:

```python
# Fungsi untuk menghitung Mean Absolute Percentage Error
# Penjelasan parameter:
# y: value aktual
# y_pred: value hasil prediksi
def mean_absolute_percentage_error(y, y_pred):
    # Konversi ke array
    actual = np.array(y)
    predicted = np.array(y_pred)

    # Perhitungan MAPE
    abs_percentage_error = np.abs((actual - predicted) / actual)
    mape = np.mean(abs_percentage_error) * 100

    return mape
```

```python
mape_test = round(mean_absolute_percentage_error(y_test, y_pred_test), 2)
mape_train = round(mean_absolute_percentage_error(y_train, y_pred_train), 2)
```

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Hasil MAPE

```
print(f"Mean absolute percentage error untuk data tes = {mape_test} %")
print(f"Mean absolute percentage error untuk data train = {mape_train} %")
```

```
Mean absolute percentage error untuk data tes = 11.72 %
Mean absolute percentage error untuk data train = 9.91 %
```

- Tampak bahwa persentase error antara harga sewa aktual dengan harga sewa hasil prediksi hanya sekitar 10 persen saja.
- Hal tersebut membuktikan bahwa model dapat memprediksi harga sewa dengan baik dan persentase errornya minim.

Kesimpulan

- Terbukti bahwa projek ini mampu mengimplementasikan tema *money* (prediksi harga) dan *investment* (investasi *real estate*) yang dapat membantu para penyewa dan investor untuk mencari bangunan yang diinginkan.
- Terbukti bahwa penggunaan model *linear regression* yang dipadukan dengan *gradient descent* berhasil diterapkan pada prediksi harga sewa bangunan di dataset ini.
- Terbukti pula bahwa model yang dibuat mampu menghasilkan hasil prediksi harga sewa yang cukup akurat. Hal tersebut sudah dibuktikan dengan berbagai jenis evaluasi yang kami lakukan.

Sekian dan Terima Kasih