# Welcome to your new notebook

# Build a machine learning model that predicts real estate prices based on historical data, using Linear Regression Model

**1. Data Collection & Preprocessing**

import pandas as pd

df = pd.read\_csv("/lakehouse/default/Files/Real estate.csv")

df.head()

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, isnull, sum, when

# Create Spark Session

spark = SparkSession.builder.getOrCreate()

# Load your dataset

df = spark.read.csv("Files/Real estate.csv", header=True, inferSchema=True)

# Count missing values for each column

df.select([sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in df.columns]).show()

**2. Feature Engineering & Data Transformation**

# **Compute price per square meter** to make comparisons easier:

df = df.withColumn("price\_per\_sq\_meter", col("Y house price of unit area") / col("X7 Property Size"))

#**Classify MRT proximity** into categories:

from pyspark.sql.functions import when

df = df.withColumn("MRT\_proximity",

           when(col("X3 distance to the nearest MRT station") < 500, "Near")

            .when(col("X3 distance to the nearest MRT station") < 2000, "Medium")

            .otherwise("Far"))

#Average price per square meter for each proximity category (Near, Medium, Far)

df.groupBy("MRT\_proximity").agg({"price\_per\_sq\_meter": "avg"}).show()

#Average house price per unit area over different transaction dates

df.groupBy("X1 transaction date").agg({"Y house price of unit area": "avg"}).show()

**3. Machine Learning Model Training**

#Split Data for Machine Learning

from pyspark.ml.feature import VectorAssembler

from pyspark.ml.regression import LinearRegression

feature\_cols = ["X2 house age", "X3 distance to the nearest MRT station", "X4 number of convenience stores", "X5 latitude", "X6 longitude", "X7 Property Size"]

assembler = VectorAssembler(inputCols=feature\_cols, outputCol="features")

df\_ml = assembler.transform(df)

display(df\_ml)

train, test = df\_ml.randomSplit([0.8, 0.2], seed=42)

#Train a Machine Learning Model

lr = LinearRegression(featuresCol="features", labelCol="Y house price of unit area")

model = lr.fit(train)

predictions = model.transform(test)

predictions.select("Y house price of unit area", "prediction").show()

**4. Model Evaluation & Feature Importance Analysis**

from pyspark.ml.evaluation import RegressionEvaluator

# Define evaluator for RMSE (Root Mean Squared Error)

rmse\_evaluator = RegressionEvaluator(labelCol="Y house price of unit area", predictionCol="prediction", metricName="rmse")

# Define evaluator for R^2 (coefficient of determination)

r2\_evaluator = RegressionEvaluator(labelCol="Y house price of unit area", predictionCol="prediction", metricName="r2")

# Compute accuracy metrics

rmse = rmse\_evaluator.evaluate(predictions)

r2 = r2\_evaluator.evaluate(predictions)

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R-squared (R²): {r2}")

Root Mean Squared Error (RMSE): 8.550167505225257

R-squared (R²): 0.6518574813100959

#Checking Feature Importance: Get feature coefficients (importance)

print("Feature Coefficients:")

for i, coef in enumerate(model.coefficients):

    print(f"{feature\_cols[i]}: {coef}")

**#Negative coefficients** → Decrease house price.

#**Positive coefficients** → Increase house price.

#**Higher absolute values** → Show stronger impact on predictions.

#Visualizing Feature Coefficients for Linear Regression

import matplotlib.pyplot as plt

import numpy as np

# Extract coefficients from the trained Linear Regression model

coefficients = model.coefficients.toArray()

feature\_names = ["X2 house age", "X3 distance to MRT", "X4 convenience stores", "X5 latitude", "X6 longitude", "X7 Property Size"]

# Sort coefficients by magnitude

sorted\_idx = np.argsort(np.abs(coefficients))[::-1]  # Sort by absolute value

sorted\_coefficients = coefficients[sorted\_idx]

sorted\_features = [feature\_names[i] for i in sorted\_idx]

# Plot feature coefficients

plt.figure(figsize=(8,6))

plt.barh(sorted\_features, sorted\_coefficients, color="lightcoral")

plt.xlabel("Coefficient Value")

plt.ylabel("Features")

plt.title("Linear Regression - Feature Coefficients")

plt.axvline(0, color='black', linestyle='dashed')  # Reference line at 0

plt.gca().invert\_yaxis()  # Highest impact at the top

plt.show()

**5. Model Predictions Visualization**

#Actual vs Predicted Prices (Scatter Plot)

import matplotlib.pyplot as plt

import pandas as pd

# Convert Spark DataFrame to Pandas for visualization

predictions\_pd = predictions.select("Y house price of unit area", "prediction").toPandas()

# Scatter plot

plt.figure(figsize=(8,6))

plt.scatter(predictions\_pd["Y house price of unit area"], predictions\_pd["prediction"], alpha=0.6, label="Predictions")

plt.plot([predictions\_pd["Y house price of unit area"].min(), predictions\_pd["Y house price of unit area"].max()],

         [predictions\_pd["Y house price of unit area"].min(), predictions\_pd["Y house price of unit area"].max()],

         color='red', linestyle='dashed', label="Perfect Predictions (Ideal Line)")

plt.xlabel("Actual House Price per Unit Area")

plt.ylabel("Predicted House Price per Unit Area")

plt.title("Actual vs Predicted Prices")

plt.legend()

plt.show()

