# Welcome to your new notebook

# Build a machine learning model that predicts real estate prices based on historical data, using Random Forest 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"))

#Check pricing trends across MRT proximity & transaction dates:

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

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

**3. Machine Learning Model Training**

#Train the Random Forest Model

from pyspark.ml.feature import VectorAssembler

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)

#plit Data into Training & Testing Sets (80% training, 20% testing):

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

#Train the Random Forest Regressor:

from pyspark.ml.regression import RandomForestRegressor

rf = RandomForestRegressor(featuresCol="features", labelCol="Y house price of unit area", numTrees=100)

model\_rf = rf.fit(train)

predictions\_rf = model\_rf.transform(test)

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

#Model Evaluation

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\_rf = rmse\_evaluator.evaluate(predictions\_rf)

r2\_rf = r2\_evaluator.evaluate(predictions\_rf)

print(f"Random Forest RMSE: {rmse\_rf}")

print(f"Random Forest R-squared (R²): {r2\_rf}")

Random Forest RMSE: 6.503947489491885

Random Forest R-squared (R²): 0.7985525040027096

#Check Feature Importance (Which Features Matter Most?)

print("Feature Importance Scores:")

for i, importance in enumerate(model\_rf.featureImportances):

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

**#Higher importance scores → Features with the strongest influence on predictions**

**#Lower importance scores → Features with less impact on price estimation**

#**No positive or negative values → Importance is based on contribution, not direction**

**#Features appearing frequently in tree splits → Indicate key variables influencing prices**

#Feature Importance Visualization

import matplotlib.pyplot as plt

import numpy as np

# Get feature importance scores from the trained model

importances = model\_rf.featureImportances.toArray()

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

# Sort features by importance

sorted\_idx = np.argsort(importances)[::-1]

sorted\_importances = importances[sorted\_idx]

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

# Plot feature importance

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

plt.barh(sorted\_features, sorted\_importances, color="skyblue")

plt.xlabel("Feature Importance Score")

plt.ylabel("Features")

plt.title("Random Forest Regression - Feature Importance")

plt.gca().invert\_yaxis()  # Highest importance 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()

