```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.5.tar.gz (317.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.2/317.2 MB 3.0 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting py4j==0.10.9.7 (from pyspark)
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl.metadata (1.5 kB)
Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.5/200.5 kB 14.7 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (pyproject.toml) ... done
  Created wheel for pyspark: filename=pyspark-3.5.5-py2.py3-none-any.whl size=317747923 sha256=746dc4949ad7f88c924365c85033436116a0
  Stored in directory: /root/.cache/pip/wheels/0c/7f/b4/0e68c6d8d89d2e582e5498ad88616c16d7c19028680e9d3840
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.7 pyspark-3.5.5
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_unixtime, count, min, max


# Initialize Spark Session
spark = SparkSession.builder.appName("MovieLensRecommendation").getOrCreate()
```

+ Code + Text

```
# Load MovieLens Dataset (33M ratings)
ratings_path = "/content/ratings.csv"
movies_path = "/content/movies.csv"
ratings = spark.read.csv(ratings_path, header=True, inferSchema=True)
movies = spark.read.csv(movies_path, header=True, inferSchema=True)


# Convert timestamps to human-readable format
ratings = ratings.withColumn("date", from_unixtime(col("timestamp")).cast("timestamp")).drop("timestamp")


# Handle missing values
ratings = ratings.dropna()


# Filter out cold-start users and movies
movies_with_enough_ratings = ratings.groupBy("movieId").agg(count("rating").alias("num_ratings"))
ratings = ratings.join(movies_with_enough_ratings, "movieId").filter(col("num_ratings") >= 10)
users_with_enough_ratings = ratings.groupBy("userId").agg(count("rating").alias("num_user_ratings"))
ratings = ratings.join(users_with_enough_ratings, "userId").filter(col("num_user_ratings") >= 10)
ratings = ratings.drop("num_ratings", "num_user_ratings")


# Normalize ratings
min_rating = ratings.agg(min("rating")).collect()[0][0]
max_rating = ratings.agg(max("rating")).collect()[0][0]
ratings = ratings.withColumn("normalized_rating", (col("rating") - min_rating) / (max_rating - min_rating))

# Save cleaned dataset as PySpark DataFrame for further model training
ratings.show(5)
```

```
+------+-------+------+-------------------+------------------+
|userId|movieId|rating|               date| normalized_rating|
+------+-------+------+-------------------+------------------+
|     1|      1|   4.0|2008-11-03 17:52:19|0.7777777777777778|
|     1|    110|   4.0|2008-11-05 06:04:46|0.7777777777777778|
|     1|    158|   4.0|2008-11-03 17:31:43|0.7777777777777778|
|     1|    260|   4.5|2008-11-03 18:00:04|0.8888888888888888|
|     1|    356|   5.0|2008-11-03 17:58:39|               1.0|
+------+-------+------+-------------------+------------------+
only showing top 5 rows
```

```
import time
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

# Start time measurement
start_time = time.time()

# Train ALS Model
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating", coldStartStrategy="drop")
als_model = als.fit(ratings)

# End time measurement
```

```
end_time = time.time()

# Print execution time
print(f"ALS Model Training Time: {end_time - start_time:.4f} seconds")
```

```
ALS Model Training Time: 32.6606 seconds
```

```
from pyspark.ml.evaluation import RegressionEvaluator

# Generate predictions
predictions = als_model.transform(ratings)

# Define evaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="prediction")
rmse_als = evaluator.evaluate(predictions)

evaluator = RegressionEvaluator(metricName="mse", labelCol="rating", predictionCol="prediction")
mse_als = evaluator.evaluate(predictions)

print(f"ALS Model -> RMSE: {rmse_als}, MSE: {mse_als}")
```

```
ALS Model -> RMSE: 0.7303235042669299, MSE: 0.5333724208847284
```

```
!pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 bytes)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl.metadata (5.2 kB)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.5.1)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow)
  Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (14 kB)
Collecting wheel<1.0,>=0.23.0 (from astunparse>=1.6.0->tensorflow)
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->ter
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflc
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflc
Requirement already satisfied: markdown>=2.6.8 in /usr/lib/python3/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.3.6)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard~=2.19.0->tensorflow)
  Downloading tensorboard_data_server-0.7.2-py3-none-manylinux_2_31_x86_64.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard~=2.19.0->tensorflow)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard~=2
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorf
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tenso
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.
Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (644.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 644.9/644.9 MB 1.5 MB/s eta 0:00:00
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-25.2.10-py2.py3-none-any.whl (30 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 3.1 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl (24.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.5/24.5 MB 73.2 MB/s eta 0:00:00
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
```

```
# Train Deep Learning Models (NCF, Autoencoders)
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Dense


import time
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Dense

# Start time measurement
start_time = time.time()

# Define input layers
user_input = Input(shape=(1,))
movie_input = Input(shape=(1,))
num_users = ratings.select("userId").distinct().count()
num_movies = ratings.select("movieId").distinct().count()

# Embedding layers for users and movies
user_embedding = Embedding(input_dim=num_users, output_dim=50)(user_input)
movie_embedding = Embedding(input_dim=num_movies, output_dim=50)(movie_input)

# Flatten the embeddings
user_vec = Flatten()(user_embedding)
movie_vec = Flatten()(movie_embedding)

# Compute dot product
dot_product = Dot(axes=1)([user_vec, movie_vec])

# Output layer
output = Dense(1, activation='linear')(dot_product)

# Build Model
ncf_model = Model([user_input, movie_input], output)
ncf_model.compile(optimizer='adam', loss='mse')

# Generate training data
train_users = np.random.randint(0, num_users, size=(100000,))
train_movies = np.random.randint(0, num_movies, size=(100000,))
train_ratings = np.random.rand(100000)

# Train Model
ncf_model.fit([train_users, train_movies], train_ratings, epochs=10, batch_size=64)

# End time measurement
end_time = time.time()

# Print execution time
print(f"NCF Model Training Time: {end_time - start_time:.4f} seconds")
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - loss: 0.1710
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - loss: 0.0641
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - loss: 0.0115
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - loss: 0.0036
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - loss: 0.0041
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - loss: 0.0067
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - loss: 0.0057
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - loss: 0.0038
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - loss: 0.0037
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - loss: 0.0041
NCF Model Training Time: 105.1360 seconds
```

```
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate predictions using the trained NCF model
predicted_ratings = ncf_model.predict([train_users, train_movies])

# Compute RMSE and MSE
mse_ncf = mean_squared_error(train_ratings, predicted_ratings)
rmse_ncf = np.sqrt(mse_ncf)
```

```
print(f"NCF Model -> RMSE: {rmse_ncf}, MSE: {mse_ncf}")
```

```
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step
NCF Model -> RMSE: 0.06327373540707314, MSE: 0.0040035655923643014
```