```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 4.4 MB/s eta 0:0
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/di
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl size
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc8a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
```

```
!pip install sparkxgb
```

```
Collecting sparkxgb
  Downloading sparkxgb-0.1.tar.gz (3.6 kB)
  Preparing metadata (setup.py) ... done
Collecting pyspark==3.1.1 (from sparkxgb)
  Downloading pyspark-3.1.1.tar.gz (212.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.3/212.3 MB 5.8 MB/s eta 0:0
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9 (from pyspark==3.1.1->sparkxgb)
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 198.6/198.6 kB 26.3 MB/s eta 0:
Building wheels for collected packages: sparkxgb, pyspark
  Building wheel for sparkxgb (setup.py) ... done
  Created wheel for sparkxgb: filename=sparkxgb-0.1-py3-none-any.whl size=562
  Stored in directory: /root/.cache/pip/wheels/b7/0c/a1/786408e13056fabeb8a72
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size
  Stored in directory: /root/.cache/pip/wheels/a0/3f/72/8efd988f9ae041f051c75
Successfully built sparkxgb pyspark
Installing collected packages: py4j, pyspark, sparkxgb
  Attempting uninstall: py4j
    Found existing installation: py4j 0.10.9.7
    Uninstalling py4j-0.10.9.7:
      Successfully uninstalled py4j-0.10.9.7
  Attempting uninstall: pyspark
    Found existing installation: pyspark 3.5.0
    Uninstalling pyspark-3.5.0:
      Successfully uninstalled pyspark-3.5.0
Successfully installed py4j-0.10.9 pyspark-3.1.1 sparkxgb-0.1
```

```
from pyspark.sql import SparkSession
from pyspark.sql.types import FloatType
from pyspark.sql.functions import col, isnan
from pyspark.ml.feature import VectorAssembler, PolynomialExpansion
from pyspark.ml.regression import RandomForestRegressor, LinearRegression, GBTR
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import RegressionEvaluator
from sparkxgb import XGBoostRegressor

# Initialize Spark session
spark = SparkSession.builder.appName("FlightDelayPred").getOrCreate()

# Load the data
data_path = "/content/drive/MyDrive/Datasets/1987.csv"  # Replace with your fil
df = spark.read.csv(data_path, header=True, inferSchema=True)

# Data type casting and handling missing values
df = df.withColumn("DepTime", col("DepTime").cast(FloatType()))
df = df.withColumn("Distance", col("Distance").cast(FloatType()))
df = df.withColumn("CRSDepTime", col("CRSDepTime").cast(FloatType()))
df = df.withColumn("Month", col("Month").cast(FloatType()))
df = df.withColumn("ArrDelay", col("ArrDelay").cast(FloatType()))
df = df.withColumn("DepDelay", col("DepDelay").cast(FloatType()))
df = df.withColumn("CRSElapsedTime", col("CRSElapsedTime").cast(FloatType()))
df = df.withColumn("DayofMonth", col("DayofMonth").cast(FloatType()))
df = df.withColumn("FlightNum", col("FlightNum").cast(FloatType()))

# Remove rows with null or NaN values in target column
df = df.filter(df.ArrDelay.isNotNull() & (~isnan(df.ArrDelay)))

# Selecting the features and target variable
features = ['Month', 'CRSDepTime', 'DepTime', 'Distance', 'DepDelay', 'CRSElaps
target = 'ArrDelay'

# VectorAssembler to combine feature columns into a single vector column
assembler = VectorAssembler(inputCols=features, outputCol="features", handleInv

# Polynomial Expansion for degree 5
polyExpansion = PolynomialExpansion(degree=5, inputCol="features", outputCol="p

# Define Linear Regression model for polynomial regression
poly_lr = LinearRegression(featuresCol="polyFeatures", labelCol=target)

# Update pipeline for Polynomial Regression
poly_pipeline = Pipeline(stages=[assembler, polyExpansion, poly_lr])

```

```python
# Define the models
rf = RandomForestRegressor(featuresCol="features", labelCol=target)
lr = LinearRegression(featuresCol="features", labelCol=target)
gbt = GBTRegressor(featuresCol="features", labelCol=target)

# Pipelines for the models
rf_pipeline = Pipeline(stages=[assembler, rf])
lr_pipeline = Pipeline(stages=[assembler, lr])
gbt_pipeline = Pipeline(stages=[assembler, gbt])

# Evaluate each model using a pipeline
models = [rf_pipeline, lr_pipeline, gbt_pipeline, poly_pipeline]

# Update the models list to include the XGBoost pipeline
models = [rf_pipeline, lr_pipeline, gbt_pipeline, poly_pipeline]

for model in models:
    # Train the model
    trained_model = model.fit(df)

    # Make predictions
    predictions = trained_model.transform(df)

    # Evaluate the model for RMSE
    rmse_evaluator = RegressionEvaluator(labelCol=target, predictionCol="predict
    rmse = rmse_evaluator.evaluate(predictions)

    # Evaluate the model for R2
    r2_evaluator = RegressionEvaluator(labelCol=target, predictionCol="predicti
    r2 = r2_evaluator.evaluate(predictions)

    # Print the RMSE and R2
    if isinstance(model.getStages()[-1], LinearRegression) and len(model.getSta
        model_name = "Polynomial Linear Regression"
    else:
        model_name = model.getStages()[-1].__class__.__name__
    print(f"{model_name} - Root Mean Squared Error (RMSE): {rmse}, R2: {r2}")

# Stop the Spark session
spark.stop()
```

```
RandomForestRegressor - Root Mean Squared Error (RMSE): 16.45410836337029, R2
LinearRegression - Root Mean Squared Error (RMSE): 15.514098363890556, R2: 0.
GBTRegressor - Root Mean Squared Error (RMSE): 14.231853301671865, R2: 0.6959
Polynomial Linear Regression - Root Mean Squared Error (RMSE): 13.17351924772
```