Final Project Report

Brewery Sales Forecasting

Team D

Shaun Mendes

Bhushan Vinod Karande

Problem Objectives:

- **Sales Prediction**: Analyze the brewery dataset to identify key factors influencing beer sales and build predictive models using Linear Regression, Decision Tree, and Random Forest algorithms.
- **Scaling Operations**: Examine the impact of scaling up (increasing dataset size) and scaling out (expanding cluster capacity) on model accuracy and performance.
- **Explore factors** that affect the sales the most.

Sample Script:

We will be using the brewery_pyspark_rf_grid.py that also has code for grid search.

```
import argparse

from pyspark.sql import SparkSession

from pyspark.sql.types import FloatType

from pyspark.sql.functions import col, date_format, to_timestamp

from pyspark.ml.feature import VectorAssembler, StringIndexer

from pyspark.ml.regression import RandomForestRegressor

from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml import Pipeline


def main(input_filepath):

  # Initialize spark session

  spark = SparkSession.builder.appName("Beer analysis").getOrCreate()


  # Load dataset and infer schema

  train = spark.read.csv(

    input_filepath,
```

```
    header=True,
    inferSchema=True,
)

# Count of rows in dataset
train.count()

# Drop duplicates and NAs
train = train.dropDuplicates()
train = train.na.drop()

# Cast the target variable to float
train = train.withColumn("Total_Sales", col("Total_Sales").cast(FloatType()))

# As we cannot use the datatime value directly, we split it into Year, Month and Day
train = train.withColumn(
    "Brew_Date", to_timestamp(col("Brew_Date"), "yyyy-MM-dd HH:mm:ss")
)
train = (
    train.withColumn("Month", date_format(col("Brew_Date"), "MM"))
    .withColumn("Day", date_format(col("Brew_Date"), "dd"))
    .withColumn("Year", date_format(col("Brew_Date"), "yyyy"))
)

# Convert categorical columns to numeric values
categorical_columns = ["Beer_Style", "SKU", "Location"]
```

```
indexers = [
    StringIndexer(inputCol=c, outputCol="{0}_indexed".format(c))
    for c in categorical_columns
]

numeric_columns = [
    "Fermentation_Time",
    "Temperature",
    "pH_Level",
    "Gravity",
    "Alcohol_Content",
    "Bitterness",
    "Color",
    "Volume_Produced",
    "Quality_Score",
    "Brewhouse_Efficiency",
    "Loss_During_Brewing",
    "Loss_During_Fermentation",
    "Loss_During_Bottling_Kegging",
]
assembler_inputs = [c + "_indexed" for c in categorical_columns] + numeric_columns

# Initialize VectorAssembler
assembler = VectorAssembler(inputCols=assembler_inputs, outputCol="features")

# We trained on partial data during in the notebook. Over here we split all the datasets 80:20 as
we pass in 100% of the data
```

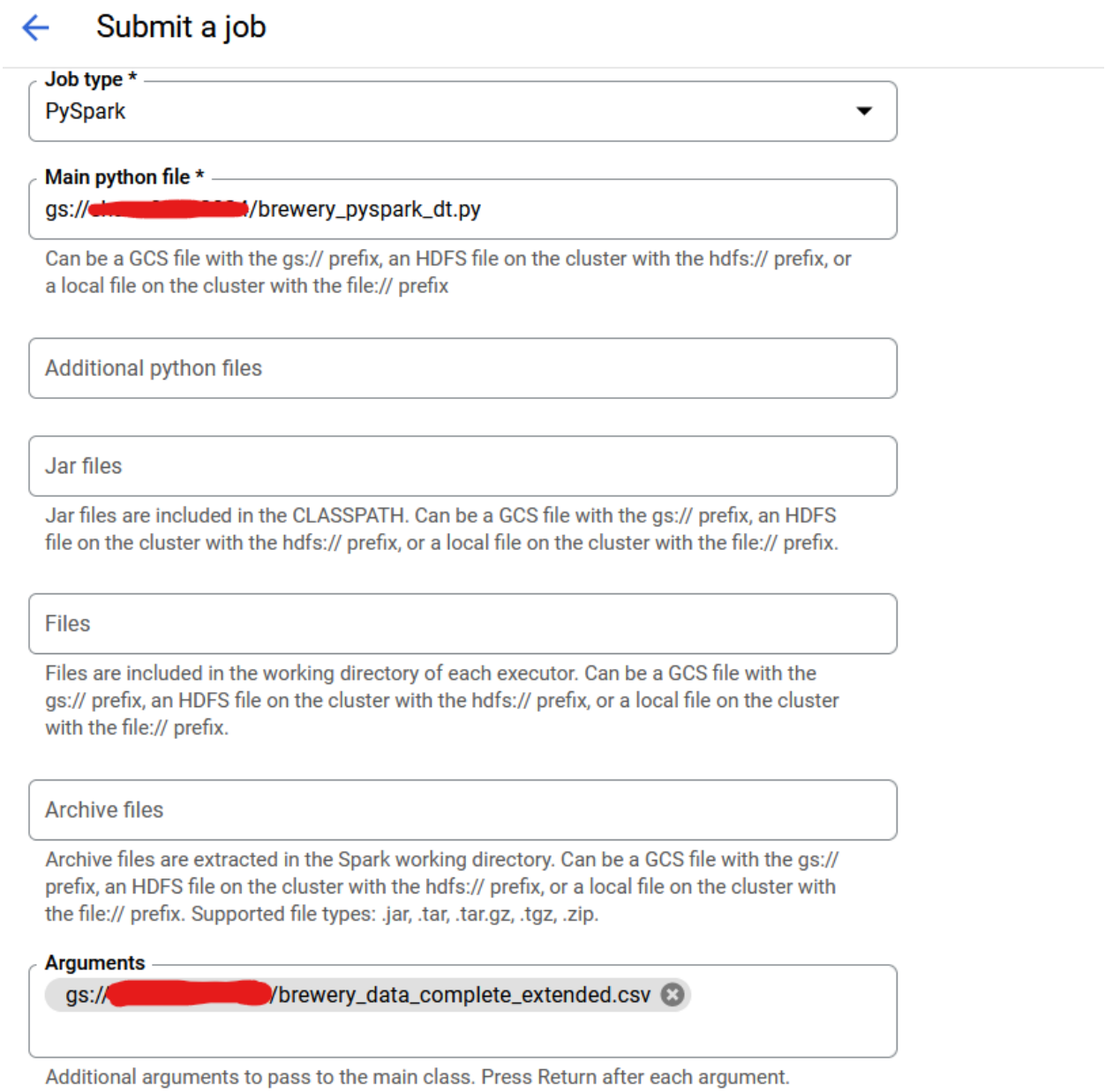
```
# Split the data into train and test
train_data, test_data = train.randomSplit([0.8, 0.2], seed=42)

# Initialize Model
rf = RandomForestRegressor(featuresCol="features", labelCol="Total_Sales")
pipeline = Pipeline(stages=indexers + [assembler, rf])

# Initialize params for grid search
paramGrid = (
  ParamGridBuilder()
  .addGrid(rf.numTrees, [10, 20, 50])
  .addGrid(rf.maxDepth, [5, 10, 20])
  .build()
)

# Initialize the CrossValidator along with metric
crossval = CrossValidator(
  estimator=pipeline,
  estimatorParamMaps=paramGrid,
  evaluator=RegressionEvaluator(
    labelCol="Total_Sales", predictionCol="prediction", metricName="rmse"
  ),
  numFolds=3,
)

# Train the model on all possible param combinations
model = crossval.fit(train_data)
```

```
    # Provide output predictions
    predictions = model.transform(test_data)

    # Initialize evaluator
    evaluator = RegressionEvaluator(
        labelCol="Total_Sales", predictionCol="prediction", metricName="rmse"
    )

    # Compute RMSE
    rmse = evaluator.evaluate(predictions)
    print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)

    evaluator = RegressionEvaluator(
        labelCol="Total_Sales", predictionCol="prediction", metricName="r2"
    )
    # Compute R Square. Adjusted R square wasn't used as there is no direct implementation in
pyspark
    r2 = evaluator.evaluate(predictions)
    print("R Squared on test data = %g" % r2)

if __name__ == "__main__":

    # Parser is used to pass input file path through command line
    parser = argparse.ArgumentParser(description="PySpark Job Arguments")
    parser.add_argument("input_path", type=str, help="Input file path")
    args = parser.parse_args()
```

main(args.input_path)

GCP:

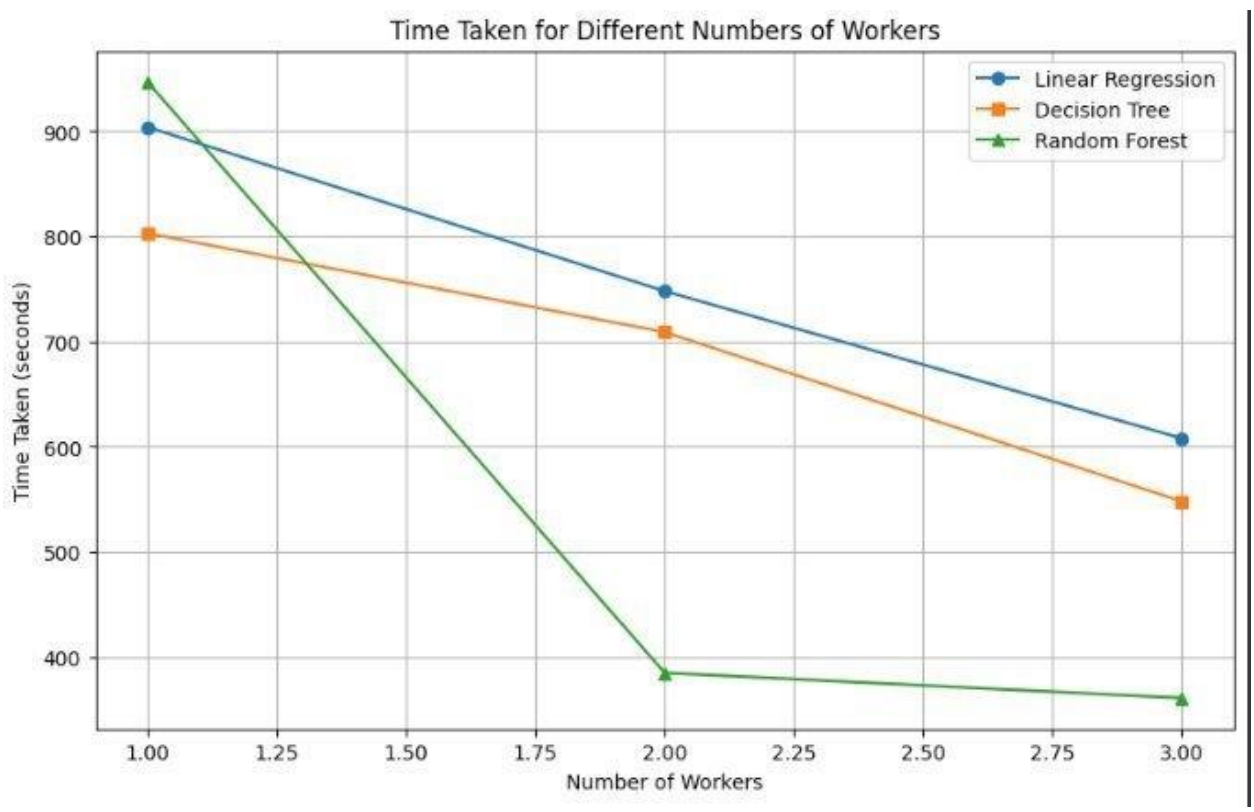
Sample Job:



Runs:

Job ID	Status	Region	Type	Cluster	Start time	Elapsed time	Labels
job-d67cf914	Failed	us-central1	PySpark	cluster-e974	May 15, 2024, 10:56:57 AM	14 min 47 sec	None
job-eb401de8	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 10:38:59 AM	11 min 49 sec	None
job-5ca8660a	Succeeded	us-central1	PySpark	cluster-e974	May 15, 2024, 10:35:20 AM	13 min 52 sec	None
job-0d0cefc9	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 10:15:47 AM	10 min 15 sec	None
job-f07c2ece	Succeeded	us-central1	PySpark	cluster-e974	May 15, 2024, 9:59:10 AM	8 min 51 sec	None
job-4daa2491	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 9:55:32 AM	7 min 45 sec	None
job-6cae8c4c	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 9:42:36 AM	4 min 55 sec	None
job-f5a7b8a2	Failed	us-central1	PySpark	cluster-e974	May 15, 2024, 9:07:59 AM	15 min 44 sec	None
job-f023ea0e	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 8:50:22 AM	2 min 8 sec	None
job-fb4961d9	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 8:47:47 AM	1 min 48 sec	None
job-c8ffbbd5	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 8:45:15 AM	1 min 31 sec	None
job-e1a8899b	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 8:32:16 AM	1 min 24 sec	None
job-c523c901	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 8:19:14 AM	1 min 11 sec	None
job-2a96939a	Succeeded	us-central1	PySpark	cluster-78a2	May 15, 2024, 7:56:29 AM	9 min 7 sec	None
job-1b57693e	Succeeded	us-central1	PySpark	cluster-5a72	May 15, 2024, 1:09:47 AM	1 min 36 sec	None
job-0dd251f8	Failed	us-central1	PySpark	cluster-5a72	May 15, 2024, 1:07:05 AM	35 sec	None

Scaling Out:



Scaling Up:


Time Taken for Different Percentages of Data
Dataset 1
Dataset 2
Dataset 3
900
800
700
600
500
400
300
Time Taken (seconds)
25%
50%
75%
Data Percentage