BDA Lab - PySpark Experiments

Experiment 10: Collaborative Filtering System using PySpark

AIM

Implement a collaborative filtering recommendation system using PySpark ALS algorithm.

PROCEDURE

1. Initialize SparkSession and load MovieLens dataset
2. Parse the rating data into DataFrame with columns: userId, movieId, rating
3. Split data into training (80%) and test (20%) sets
4. Create ALS model with specified parameters (rank, maxIter, regParam)
5. Train the ALS model on training data
6. Generate predictions on test data
7. Evaluate model using RMSE metric
8. Generate top-N movie recommendations for users

SAMPLE INPUT

```
userId,movieId,rating,timestamp
1,1,5.0,874965758
1,2,3.0,876893171
2,1,4.0,878542960
```

SAMPLE OUTPUT

```
RMSE: 0.95
Top 5 recommendations for user 1:
- Movie ID 50: Predicted Rating 4.8
- Movie ID 181: Predicted Rating 4.6
- Movie ID 294: Predicted Rating 4.5
```

RESULT

Successfully implemented collaborative filtering with ALS achieving RMSE of 0.95.

Experiment 11: Classification Algorithms Comparison using PySpark

AIM

Compare Logistic Regression, SVM, and Decision Tree classifiers on classification data using PySpark.

PROCEDURE

1. Initialize SparkSession and load Iris dataset
2. Create feature vector using VectorAssembler
3. Split data into training (70%) and test (30%) sets
4. Train three models: LogisticRegression, LinearSVC, DecisionTreeClassifier
5. Generate predictions for each model on test data
6. Calculate precision, recall, and F1-score for each algorithm
7. Create comparison chart/graph of metrics
8. Identify best performing algorithm

SAMPLE INPUT

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
7.0,3.2,4.7,1.4,versicolor
```

SAMPLE OUTPUT

```
Algorithm Comparison:
Logistic Regression - Precision: 0.96, Recall: 0.96, F1: 0.96
SVM - Precision: 0.95, Recall: 0.95, F1: 0.95
Decision Tree - Precision: 0.93, Recall: 0.93, F1: 0.93
```

RESULT

Logistic Regression achieved highest performance with F1-score of 0.96. The comparison has been successfully completed.

Experiment 12: K-Means Clustering using PySpark

AIM

Implement K-Means clustering algorithm using PySpark MLlib on Iris dataset.

PROCEDURE

1. Initialize SparkSession and load Iris dataset
2. Create feature vector using VectorAssembler (exclude target column)
3. Determine optimal number of clusters using elbow method
4. Initialize KMeans model with k=3 clusters
5. Train the clustering model on feature vectors
6. Generate cluster predictions for all data points
7. Evaluate clustering using silhouette score
8. Visualize clusters and centroids

SAMPLE INPUT

```
sepal_length,sepal_width,petal_length,petal_width
5.1,3.5,1.4,0.2
4.9,3.0,1.4,0.2
7.0,3.2,4.7,1.4
```

SAMPLE OUTPUT

```
Optimal clusters: 3
Silhouette Score: 0.72
Cluster Centers:
Cluster 0: [5.01, 3.42, 1.46, 0.24]
Cluster 1: [5.90, 2.75, 4.39, 1.43]
Cluster 2: [6.85, 3.07, 5.74, 2.07]
```

RESULT

The clustering using K-Means Clustering on Iris data into 3 groups has been successfully completed with a silhouette score of 0.72.

> *Ahmed Baari. 126156004.*