Class Note:

Subject : .....BDT.....

Faculty : .....V. Divya.....

Topic: Large scale ML with Spark - Basic statistics.

Unit No. : 5
Lecture No.: 1
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 1

Basic statistics in PySpark for Large scale Machine Learning:

- Basic statistics are foundational for exploring and understanding large datasets before building machine learning models, and they play a critical role in data preprocessing, feature engineering and selecting appropriate algorithms.
- Here's a rundown of the main basic statistics functions that are often used in PySpark to handle large datasets:

(i) Descriptive statistics:

- It provide essential information about the distribution and ~~provide~~ range of features such as mean, variance, minimum and maximum values.
- This provides:
  - → Mean
  - → Variance
  - → Min/Max

(ii) Correlation:-

- The correlation between features can help determine relationships and dependencies, which can be useful for feature selection and engineering.

- The correlation matrix can reveal whether certain features are highly correlated, which may suggest redundancy or collinearity.

iii, Covariance:

- The covariance matrix indicates how much two features vary together, which is useful for understanding the relationships between them.

iv, Sampling and Stratified Sampling:

- Sampling can help with quick exploratory analysis or for generating balanced training/test splits in machine learning.
- Stratified sampling is especially useful for maintaining class balance in classification problems.

v, Standardization:

- Standardizing data (scaling to have zero mean and unit variance) is important for algorithms that are sensitive to feature scaling.

vi, Hypothesis Testing:

- PySpark offers statistical tests to evaluate the independence of features, which is helpful for feature selection.
- The Chi-square test is commonly used for testing independence with categorical data.

vii, Data Splitting for Train/Test sets:-

- Splitting data into training and testing sets is crucial for evaluating model performance and avoiding overfitting.

viii, Handling Missing data:

- Handling missing values is a key part of data preprocessing. PySpark provides fill, drop and Imputer functionalities.

Class Note:

Subject : .....B.D.T.....

Faculty : .....V.Divya.....

Topic: .....Data Sources.....

Unit No. : 5
Lecture No.: 2
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 2

Data sources:

- In Spark, various data sources can be used for large scale machine learning, each catering to different storage types, data access methods and usecases.
- These datasources enable Spark to read and write massive datasets stored across distributed systems, allowing for efficient data processing in machine learning pipelines.
- Here are some commonly used data sources in Spark that support large scale machine learning applications:

(i) HDFS:-

- HDFS is a widely used storage layer in big data ecosystems. It is designed to handle massive amounts of data and offers fault tolerance and scalability.

→ Common Usecases: Batch data processing, large scale data storage.

→ Setup: Spark typically reads from and writes to HDFS by specifying hdfs://<path>.

(ii) Amazon S3:

- It is an object storage service commonly used in Spark deployments on AWS. S3 provides virtually unlimited storage capacity and high durability.

→ Common usecases: Storing raw or processed data for Spark jobs, backup and archiving.
→ Setup: Use s3a:// for optimal performance.

iii, GCS:
- is an object storage service used in Google Cloud environments, especially with Google Cloud Dataproc clusters running Spark.
→ Common Use cases: Data storage for ML models and ETL pipelines, backup.
→ Setup: Access GCS with gs://

iv, Azure Blob Storage:
- It is a scalable storage solution from Microsoft, often used with Spark in Azure databricks.
→ Common Usecases: Data storage for analytics, backup and archiving.
→ Setup: Access it with wasbs:// & abfss://

v, Delta lake:
- It is an open source storage layer that brings ACID transactions and schema enforcement to Spark.
- It enables scalable data lakes with batch and streaming data in a unified format.
→ Common Usecases: Machine Learning data lakes, streaming data processing, time travel.
→ Setup: Access Delta tables using Delta format.

vi, Apache Cassandra:
- It is a NoSQL distributed database designed for scalability and high availability.
- Spark's Cassandra connector allows for direct reads and writes.

Class Note:

Subject : .....BDT.....

Faculty : .....V. Divya.....

Topic: ...Data sources, Pipelines.

Unit No. : 5
Lecture No.: 2, 3
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:

→ Common Usecases: Real time machine learning features, distributed data storage.

→ Setup: Access with Spark Cassandra Connector.

vii, Apache HBase:-

- Apache HBase is a distributed, scalable, NoSQL database built on top of HDFS. It is used for low-latency storage and retrieval of large datasets.

→ Common Usecases: Serving machine learning model predictions, real time feature storage.

→ Setup: Access with HBase connector.

Pipelines:

- These help automate and streamline the workflow from data ingestion to model training, tuning and deployment.
- Spark MLlib and Spark ML provide a structured approach to building machine learning pipelines by chaining together various stages, such as data preprocessing, feature engineering, model training and evaluation.

- Here are the key stages and components used to build efficient large scale ML pipelines in Spark:-

i, Data Ingestion:

- It is the first stage, where raw data is loaded from various data sources into Spark for processing.
- Depending on the usecase, the data might come from batch or streaming sources.

→ Batch data sources: Data Lakes (HDFS, S3), data warehouses (Hive), & databases (via JDBC)

→ Streaming data sources: Real time sources like Kafka, Kinesis or structured streaming from file sources.

ii, Data Preprocessing and Cleaning:

- Raw data often needs cleaning and transformation to be ready for model training.
- Preprocessing might include handling missing values, filtering outliers, and standardizing formats.
- In PySpark, 'Dataframe' transformations and Spark ML transformers are commonly used for this.

→ Handling missing values: Use "fillna" or "Imputer"

→ Data Transformation: Spark ML transformers like "StringIndexer", VectorAssembler (for assembling features), StandardScaler and others.

iii, Feature Engineering:

- In feature engineering, we transform and generate additional features to improve model performance.
- This stage includes:

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Pipelines

Unit No. : 5
Lecture No.: 3
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 74

→ Scaling and Normalization: Standardize features to improve convergence in training
→ Feature Selection: Select relevant features to reduce dimensionality
→ Text Processing: Convert text data into numeric features.

iv) Model Training:

- In Spark, machine learning models are created with algorithms from Spark MLlib, including regression, classification, clustering and collaborative filtering models.
- The pipeline structure allows models to be part of stages.

→ Common Algorithms: Linear Regression, Logistic Regression, Decision trees, Random Forest, Gradient Boosted trees
→ Pipelining Models: Combine pretrained stages with a model estimator for a single cohesive workflow.

v) Hyperparameter Tuning:

- It is crucial in large scale ML pipelines to optimize model performance.
- Spark MLlib offers CrossValidator and TrainValidationSplit to facilitate hyperparameter tuning across distributed data.

→ Cross Validation: Splits data into training and test sets, trains the model on multiple configurations, and evaluates the best.

→ Grid Search: Allows specifying hyperparameter to test different model configurations in Spark.

vi, Model Evaluation:

- After training, evaluate model performance on a separate test set using evaluation metrics specific to the problem.

→ Common metrics: Accuracy, Precision, Recall, f1 score for classification

→ Evaluators: Use Spark's "BinaryClassificationEvaluator", "MultiClass classificationEvaluator" & "RegressionEvaluator".

vii, Deployment:

- For Real time applications, trained models can be deployed to score new data in production using Spark structured streaming or batch processing with saved model artifacts.

→ Saving Models: Use model.save() to store the model for future loading.

→ Streaming Predictions: Leverage Spark's "structured streaming" to score streaming data in realtime.

viii, Monitoring and maintenance

- In production, monitoring model performance and retraining when necessary is essential.

- This may include:

→ Drift detection: Detect shifts in data distribution or performance metrics.

→ Model Retraining: Retrain models periodically with new data or upon detecting drift.

→ Logging and Metrics: Use monitoring tools to track key metrics such as prediction latency, accuracy & resource usage.

Class Note:

Subject : ....BDT.....

Faculty : ....V.Divya.....

Topic: .....Extracting.....

Unit No. : 5
Lecture No.: 4
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:

Extraction:-

- When working with large scale ML and data extraction, PySpark can be powerful tool due to its distributed data processing capabilities.
- Here's an outline of how to perform data extraction at scale using PySpark and prepare it for mlc learning:

(i) Setup PySpark environment:

- Install PySpark if not already installed.
- Import necessary libraries
- Initialize a Spark session

(ii) Load Data efficiently:-

- PySpark allows loading data from various sources, like HDFS, Amazon S3, & local file systems.

(iii) Data cleaning and transformation:

- Use PySpark's SQL dataframe functions for cleaning and transforming large datasets.

(iv) Feature Engineering:-

- Feature engineering is critical in ML, and PySpark provides various utilities for it:

→ VectorAssembler: Combine multiple columns into a single

feature vector.

→ StandardScaler: Standardize feature for better ML model performance.

v, Train test Split:-

- Split the data into training and testing sets to evaluate your ML model

vi, Define and train the ML model:-

- Choose a model, such as a Linear Regression model for training.

vii, Evaluate the Model:-

- Use evaluation metrics to assess the model's performance.

viii, Optimize and Scale up:-

- Hyperparameter Tuning: Use CrossValidator or TrainValidationSplit for tuning model parameters.
- Pipeline: Use pipelines to streamline preprocessing and training steps, which is useful when running experiments at scale.

Class Note:

Subject : .....BDT.....

Faculty : .....V. Divya.....

Topic: .....Transforming in Pyspark, Selecting in PySpark.

Unit No. : 5
Lecture No.: 5, 6
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 6

Feature transformation:

- Transforming features is a preprocessing step to convert raw data into a format that is suitable for machine learning models.

i) Vector Assembler:
- Combines multiple columns into a single feature vector, which is essential for machine learning models that expect a single features column.

ii) Standard Scaler:
- Standardizes features to have a mean of zero and a standard deviation of one, which can help models converge faster.

iii) MinMax Scaler:
- Scales each feature to a specific range, usually [0, 1]

iv) One Hot Encoder:
- Converts categorical features into a one-hot encoded vector.

v) PolynomialExpansion:
- Generates polynomial features of input columns, which can help in capturing non-linear relationships.

Feature Selection:

- Feature selection reduces the no. of features to avoid overfitting, improve performance, and reduce computational costs.

(i) Chi-Square Selector:

- Selects features based on chi-square statistics, useful for categorical data.

(ii) Variance Threshold Selector:

- Removes features with low variance, which are less likely to provide valuable information.

(iii) Principal Component Analysis (PCA):

- Reduces the feature set by projecting it into a lower-dimensional space while preserving variance.

(iv) Rformula:

- Rformula is a high level API for feature selection that simplifies transformations by selecting features and target variables using a formula syntax.

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Classification and Regression.

Unit No. : 5
Lecture No.: 7
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 7

Classification & Regression in PySpark:

(i) Setting up PySpark environment:
- Import necessary libraries and initialize a Spark session.

(ii) Data preparation:
- Load your data and prepare it for modeling. For MLlib, features should be in a single vector column.

(iii) Splitting data into training and test sets:
- To evaluate your model, split the data into training and test sets.

(iv) Classification Algorithms:
- PySpark provides several classifiers that can scale well on large datasets, such as Logistic Regression, Random Forest and Gradient Boosted Trees.

(a) Logistic Regression:
- Logistic regression is a linear model commonly used for binary classification.

(b) Random Forest Classifier:
- Random Forest is an ensemble method that combines multiple decision trees and works well on large datasets.

c, Gradient Boosted Trees Classifier:
- GBT is another ensemble technique, but it builds trees sequentially to reduce errors.

v, Regression Algorithms:
- PySpark's MLlib also supports regression models such as Linear Regression, Decision Trees, and Random Forests.

a, Linear Regression:
- Linear Regression is a simple yet effective model for predicting continuous values.

b, Random forest Regressor:
- The Random forest Regressor is useful for capturing complex relationships in data.

c, Gradient Boosted Trees Regressor:-
- Like the GBT classifier, GBT regressor iteratively builds trees to minimize errors in regression tasks.

vi, Evaluating models:
- For classification, we can use evaluation metrics like AUC and accuracy.
- For regression, use RMSE or R-squared.

a, Classification evaluation

b, Regression evaluation

vii, Using Pipelines for workflow automation:
- To streamline the process of data preprocessing model training, and evaluation, use a pipeline.
- Pipelines are especially helpful when iterating on models.

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Clustering

Unit No. : 5
Lecture No.: 8
Link to Session
Planner (SP) S.No. ..... of SP
Date Conducted:
Page No. 8

Clustering:

- Clustering in large scale machine learning with PySpark is effective for grouping data without labeled outputs, such as segmenting customers, discovering patterns in datasets & anomaly detection.
- PySpark's MLlib supports scalable clustering algorithms, most notably K-Means and Gaussian Mixture Models (GMM), designed to handle distributed data.

(i) Setting up PySpark environment:

- Start by importing necessary libraries and initializing a Spark session.

(ii) Load and Prepare data:

- Load our dataset and prepare it by combining features into a single vector, as clustering algorithms in PySpark require the features to be in a single "features" column.

(iii) Feature scaling:

- Scaling features can help improve clustering performance by ensuring that features contribute equally to the distance calculations.

iv, Clustering Algorithms:

a, K-Means Clustering:

- K-Means is one of the most commonly used clustering algorithms due to its simplicity and scalability.
- K-Means in PySpark uses the parallelized Lloyd's algorithm to scale on large datasets.

b, Gaussian Mixture Model (GMM):

- These are useful when clusters may overlap, as they treat each cluster as a Gaussian distribution and calculate probabilities for each point to belong to a cluster.

v, Evaluating clustering performance:

- After clustering, evaluate the model's performance.
- for K-Means, the Silhouette Score is a common metric that measures how close each point in one cluster is to points in neighboring clusters.

vi, Hyperparameter tuning for optimal number of clusters (K):

- Choosing the right number of clusters 'k' is critical in clustering.
- One approach is to compute the Silhouette Score for different values of 'k' and choose the value with the best score.
- Alternatively, the Elbow method can be used by plotting the cost Ewithin t for each value of 'k'.

vii, Using pipelines to automate clustering workflow:

- Create a pipeline to automate preprocessing, clustering and evaluation.
- This is useful when iterating through models or deploymenting a clustering workflow.

Class Note:

Subject : .....B.D.T.....

Faculty : ....V. Divya.....

Topic: ...Collaborative filtering

Unit No. : 5
Lecture No.: 9
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 9

viii, Persisting and saving the clustering model:-
- For large scale production applications, saving models and clustering results is crucial for model deployment and reproducibility.

ix, Optimizing performance in Large scale clustering:-
- Caching: Cache the dataframe if it's reused multiple times to reduce recomputation.
- Partitioning: Properly partition data to optimize resource utilization, especially if working in a cluster environment.
- Resource allocation:- Configure Spark executors and memory based on dataset size and available cluster resources.

Collaborative filtering:-
- Collaborative filtering is a popular recommendation technique, often used in recommendation engines to provide personalized recommendations based on user behavior.
- In large scale mlc learning, collaborative filtering is typically implemented using matrix factorization methods like Alternating Least Squares (ALS).
- PySpark's MLlib library includes a scalable implementation

of ALS, making it a suitable choice for processing large datasets distributed across multiple nodes.

Step-by-step guide to implement Collaborative Filtering in PySpark:

i) Setup and import libraries:
- We will need PySpark installed and set up to start. Here and import for implementing ALS in PySpark.

ii) Initialize Spark session:
Create a Spark session to interact with PySpark.

iii) Load and Preprocess the data:
- Load the dataset, which should contain atleast user-item interaction data, commonly in the form of user ID, item ID and a rating or interaction score.
- Ensure that the dataset is preprocessed, meaning that all user IDs and item IDs are numeric.
- We may also want to remove duplicates and handle missing values as necessary.

iv) Split the data:
- Divide the data into training and test sets to evaluate the model.

v) Train the ALS model:
- Set up the ALS model with hyperparameters for rank, regularization and iterations
  → maxIter: No. of iterations
  → regParam: Regularization parameter to prevent overfitting.
  → coldStartStrategy: Drop NaN values in predictions.

Class Note:

Subject : .....BDT.....

Faculty : ....V. Divya.....

Topic: Collaborative filtering, Frequent Pattern Mining.

Unit No. : 5
Lecture No.: 9,10
Link to Session Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 10

vi, Evaluate the model:-
- Use a regression evaluator to measure the model's performance on the test data.
- Root mean Squared Error (RSME) is a common metric for collaborative filtering.

vii, Generate Recommendations:-
- After evaluating, we can generate recommendations for users.
- ALS can provide top-N recommendations for each user or item.

viii, Tuning Hyperparameters:-
- We can further tune parameters using a grid search or cross-validation to find the best rank, maxIter, and regParam values.
- This process can significantly improve recommendation quality but is computationally intensive.

ix, Stop the spark session:-
- End the session after training and evaluation to free resources

Frequent Pattern Mining:

- It is an essential technique in machine learning and data mining, used to discover patterns, associations & correlations among large datasets.
- PySpark's MLlib provides tools for frequent pattern mining that scale well to handle large datasets.
- One of the most common algorithms for this purpose is the FP-Growth algorithm, which is efficient for discovering frequent itemsets in large scale datasets.

Step-by-step guide to implement Frequent Pattern mining in PySpark:

i) Setup and import libraries:
- Start by setting up your PySpark environment and importing the necessary libraries for frequent pattern mining.

ii) Initialize a spark session:
- Create a spark session to handle the data processing.

iii) Load and Preprocess data:
- Load dataset. and the data should have atleast one column representing a collection of items for each transaction.
- Each transaction is typically represented as a row with a list of items.

iv) Train the FP-Growth model:
- FP-Growth requires specifying the minimum of support and confidence thresholds:

→ minSupport: The minimum frequency a pattern must appear to be considered "frequent".

→ minConfidence: The minimum confidence for generating association rules.

Class Note:

Subject : .....BDT.....

Faculty : ....V.Divya.....

Topic: ....frequent Pattern Mining & pattern mining steps

Unit No. : 5
Lecture No.: 10, 11
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 11

v, Extract frequent Itemsets:
- Once trained, the model provides frequent itemsets that meet the specified minimum support.

vi, - The freqItemsets Dataframe will contain columns for each itemset and their corresponding support counts.

vi, Generate Association Rules:
- The fP-Growth model can also generate association rules. and these rules represent strong relationships among items in the dataset, given the specified confidence threshold.
- This dataframe will contain antecedent, consequent and confidence columns for each rule, showing which itemsets lead to others with sufficient confidence.

vii, Transform New data:
- We can use the trained model to find items that are likely to cooccur with items in new trasactions.
- This allows the model to recommend associated items for new inputs.
- The transform function will return the input transactions with an additional column for potential recommendations based on frequent patterns found in training.

viii) Stop the Spark session:

- Close the Spark session to free resources.

Class Note:

Subject : ...BDT.....

Faculty : ...V.Divya.....

Topic: ...Model selection and Tuning

Unit No. : 5
Lecture No.: 12,13.
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 12

Model Selection:

- Model selection in large scale machine learning with PySpark involves choosing the best model and optimal hyperparameters for a given task, balancing accuracy and computational efficiency.
- PySpark's MLlib provides tools like CrossValidator and TrainValidationSplit for hyperparameter tuning and hyperparameter grids allow efficient exploration of parameters spaces in a distributed setting, making PySpark ideal for large scale applications.

Steps for model selection and tuning in PySpark:

i) Set up and import libraries:

- To start, we need to set up the PySpark environment and import the necessary libraries for machine learning, hyperparameter tuning and evaluation.

ii) Initialize the spark session:

- Create a spark session, which provides the context needed for distributed data processing.

(iii) Load and prepare data:

- Load the dataset and conduct any necessary preprocessing, including handling missing values, encoding categorical variables, and vectorizing features.
- Data preparation is essential for improving model performance and consistency.

(iv) Define the model and pipeline:

- Choose the machine learning algorithm that fits our problem.
- In this We use "RandomForestClassifier" for binary classification.
- The "pipeline" allows chaining data transformations and the model, making it easier to tune and reuse.

(v) Define a Hyperparameter grid:

- Use a "ParamGridBuilder" to create a grid of hyperparameters to search across.
- Common hyperparameters to tune for "RandomForestClassifier" include:
  - → numTrees: The number of trees in the forest.
  - → maxDepth: The maximum depth of trees.
  - → subsamplingRate: The fraction of training data used for each tree

(vi) Define an Evaluator:

- Select an evaluator based on model's objective.
- For classification, AUC (Area under the ROC curve) is commonly used, while regression tasks often use Root Mean Squared Error (RMSE).

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Model selection and Tuning.

Unit No. : 5
Lecture No.: 13
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 13

vii, Setup cross-validation or train-validation split:-
- In large scale applications, two main methods are available for model selection and tuning:

→ CrossValidator: Performs k-fold cross-validation and evaluates each combination of parameters.

→ TrainValidationSplit: Randomly splits the data into training and validation sets and evaluates each combination of parameters, providing a faster alternative to cross-validation.

viii, Train the model and perform Hyperparameter tuning:-
- Fit the model to the training data using either crossval.fit() or train_val_split.fit().
- This process iterates through the parameter grid and evaluates each combination on the validation set, returning the best model.

ix, Evaluate the Best model on Test Data:
- Retrieve the best model from tuning process and evaluate it on the test set to check its performance on unseen data.

(x) Inspect the Best Model's HyperParameters:-

- After model selection, we can check the hyperparameters of the best model found during tuning.

xi) Stop the Spark session:

End the Spark session once trainging and evaluation are complete to free up resources.