****

**SIMATS ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL**

**SCIENCES – 602105**

**A CAPSTONE PROJECT REPORT**

**CSA1583- Cloud Computing and Big Data Analytics Using Cloud Federation**

Submitted in the partial fulfilment for the award of the degree of

**BACHELORE OF ENGINEERING**

**IN**

**COMPUTER SCIENCE ENGINEERING**

**Design and implement an HR management system through any open cloud platform.**

**A PROJECT REPORT**

**Done by:**

A. DHANUSH (192210588)

**Supervisor**

Dr. Chenni Kumaran

**DECLARATION**

I am A. Dhanush student of computer science and Engineering, Saveetha School of Engineering, saveetha Institute of Medical and Technical Sciences, Chennai. hereby declare that the work presented in the capstone project work.

**A. Dhanush (192210588)**

**Date:**

**Day:**

**CERTIFICATE**

**This is to certify that the project entitled “Design and implement an HR management system through any open cloud platform.” Submitted by A.DHANUSH has been carried out under our supervision. The project has been submitted as per the requirements in the current semester of B. Tech Information Technology.**

Faculty-in-Charge

Dr. chenni Kumaran

**TABLE OF CONTENT**

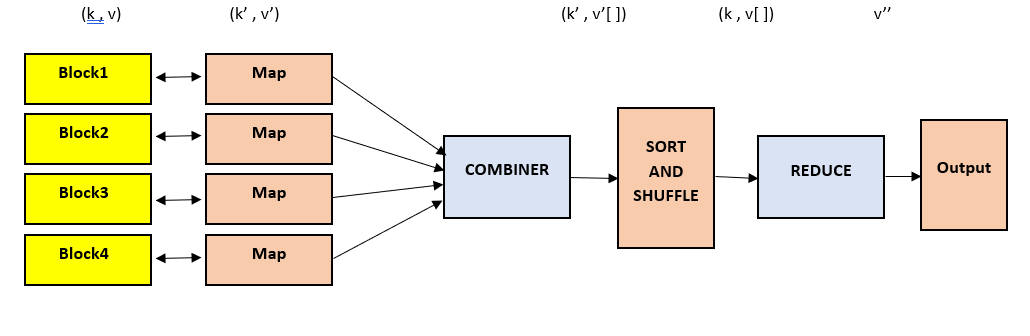
|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Components** | **Marks (100)** |
| **1** | **Problem Statement** | **(5)** |
| **2** | **Proposed Design Work** | **(20)** |
|  | * Identifying Key Components | (5) |
|  | * Functionality | (5) |
|  | * Matchmaking Architectural Design | (10) |
| **3** | **Best cloud node prediction Design** | **(20)** |
|  | * Layout | (10) |
|  | * User-Friendly | (5) |
|  | * Resource Selection | (5) |
| **4** | **Program / Coding** | **(20)** |
|  | * Language code | (5) |
|  | * Algorithm/Program | (10) |
|  | * Execution | (5) |
| **5** | **Implementation** | **(20)** |
|  | * Connecting the Components in Cloud | (5) |

**Design and implement an HR management system through any open cloud platform.**

**Problem Statement:**

**In the realm of big data analytics, the need for efficient classification models is paramount. Traditional machine learning approaches face challenges with scalability and performance when dealing with large datasets. This project aims to develop a scalable MapReduce-based classification model that incorporates feature subset selection and hyper-parameter tuning using a deep belief network (DBN).**

**Proposed Design Work :**



**Identifying Key Components**

**1. \*MapReduce Framework Integration\*:** Implement a distributed computing framework using MapReduce to handle the computational challenges posed by large-scale datasets. This framework will enable efficient parallel processing for feature selection and model training.

**2. \*Feature Subset Selection\*:** Explore and implement techniques for identifying and selecting relevant features from high-dimensional datasets. The goal is to enhance model performance and reduce computational overhead by focusing on the most informative features.

**3. \*Deep Belief Network (DBN) Construction\*:** Construct and optimize a DBN for classification tasks. DBNs are chosen for their ability to learn complex hierarchical representations of data, which is beneficial for capturing intricate **patterns in big data.**

**4. \*Hyper-parameter Tuning**\*: Employ strategies for hyper-parameter optimization to enhance the DBN's performance. This includes techniques such as grid search, random search, or Bayesian optimization to fine-tune model parameters effectively**.**

**5. \*Evaluation and Comparison**\*: Evaluate the proposed MapReduce-based DBN model against baseline classifiers and other state-of-the-art approaches. Performance metrics such as accuracy, precision, recall, and F1-score will be used for comprehensive evaluation.

**6. \*Deployment Considerations\*:** Discuss practical considerations for deploying the developed model in real-world big data environments. This includes scalability, computational efficiency, and integration with existing big data infrastructure**.**

**Functionality in MapReduce-based big data classification model:**

**1. Data Ingestion and Preprocessing**

**\*Data Loading\*:** Use MapReduce to ingest large volumes of data from distributed storage systems such as HDFS (Hadoop Distributed File System**).**

\* Implement MapReduce tasks to clean the data, handling missing values, outliers, and inconsistent data entries**.**

**\*Data Transformation\*:** Perform transformations such as normalization or encoding categorical variables in a distributed manner using MapReduce

**2. Feature Subset Selection**

**\*Feature Extraction**\*: Use MapReduce to extract features from raw data, especially useful for complex or large datasets**.**

**\*Feature Selection Algorithms\*:**

**\*Filter Methods**\*: Use Implement distributed algorithms for selecting a subset of relevant features. Techniques could include statistical measures (e.g., correlation, chi-square) to score features and select the most significant ones.

**\*Wrapper Methods**\*: Apply MapReduce to evaluate feature subsets based on the performance of a classification model**.**

**\*Embedded Methods\*:** Integrate feature selection directly into the MapReduce training process of models like decision trees or LASSO regression**.**

**3. Distributed Model Training**

**\*Deep Belief Network (DBN) Training**\*: Utilize MapReduce to train a DBN in a distributed fashion. Each Mapper could handle a portion of the training data, and Reducers could aggregate the updates to the network’s weights**.**

**- \*Incremental Learning\*:** Enable the model to update its parameters incrementally as new data arrives, improving scalability and adaptability**.**

**\*Parallelism\*:** Exploit the parallel nature of MapReduce to divide the DBN training workload across multiple nodes, reducing training time**.**

**4. Hyper-Parameter Tuning**

**\*Distributed Search**\*: Implement distributed versions of hyper-parameter tuning techniques (e.g., grid search, random search) using MapReduce to evaluate different configurations of the DBN across multiple nodes**.**

**\*Model Evaluation\*:** Use MapReduce to perform cross-validation and other evaluation techniques on distributed data to identify the optimal hyper-parameters.

**5. Classification and Prediction-** \*Distributed Classification\*: Apply the trained DBN to classify new data using MapReduce, where Mappers distribute the workload of classifying different data segments.

**6. Model Evaluation and Metrics Calculation**

**\*Performance Metrics**\*: Implement MapReduce jobs to calculate performance metrics like accuracy, precision, recall, and F1-score across large datasets.

**\*Confusion Matrix\*:**

**\*Scalability Testing\*:** Evaluate the Use MapReduce to generate and analyze the confusion matrix for multi-class classification problems model’s performance under different data volumes to ensure scalability.

**7. Data Output and Visualization**

**\*Result Aggregation**\*: Use Reducers to consolidate classification results from distributed Mappers and store them back into a distributed file system**.**

**- \*Visualization**\*: Integrate tools or frameworks (e.g., Apache Zeppelin, Tableau) to visualize the classification results and performance metrics.

**8. Deployment and Integration**

**- \*Model Deployment**\*: Develop mechanisms for deploying the trained DBN model into a production environment that supports distributed data processing**.**

**- \*API Integration\*:** Provide APIs or interfaces for other systems to interact with the model, facilitating its use in broader applications**.**

**9. Fault Tolerance and Efficiency**

**\*Checkpointing**\*: Implement checkpointing to save the model’s state periodically, ensuring that training can resume from a recent point in case of failures.

**- \*Resource Management\*:** Optimize resource usage in the MapReduce framework to improve efficiency and reduce computation costs**.**

**BEST CLOUD NODE PREDICTION DESIGN:**

**Layout Design:**

**Layout Design for MapReduce-based Big Data Classification Model**

1. \***Data Ingestion and Preprocessing\***

**\*Map**: \* Load, clean, and transform data.

\***Reduce: \*** Aggregate processed data.

**2. \*Feature Subset Selection\***

\***Map: \*** Score and evaluate features.

\***Reduce**: \* Select top features.

**3. \*DBN Training**\*

**\*Map: \*** Train DBN on data partitions, compute updates.

\***Reduce:** \* Aggregate updates, synchronize model.

4**. \*Hyper-Parameter Tuning\***

\***Map: \*** Evaluate different hyper-parameter sets.

**\*Reduce: \*** Aggregate results, select best parameters.

5. \*Classification and Prediction\*

**\*Map**: \* Classify new data using the trained DBN.

\***Reduce: \*** Aggregate predictions.

6. \*Model Evaluation\*

**\*Map: \*** Calculate local metrics.

\***Reduce:** \* Combine metrics for overall evaluation.

**User-Friendly Layout Design for MapReduce-Based Big Data Classification Model:**

1**. \*Data Ingestion and Preprocessing\***

**\*Map: \*** Load large datasets, clean up inconsistencies, and transform data formats.

\***Reduce**: \* Merge the processed data into a unified, clean set.

2. **\*Feature Subset Selection\***

**\*Map: \*** Score and assess each feature's importance.

**\*Reduce**: \* Select and keep only the most valuable features for the model.

3**. \*Deep Belief Network (DBN) Training\***

**\*Map: \*** Train parts of the DBN on chunks of data, adjusting weights locally.

**\*Reduce: \*** Combine weight updates to create a unified, trained model.

4. **\*Hyper-Parameter Tuning\***

\***Map: \*** Test different configurations of model parameters.

**\*Reduce: \*** Find and choose the best-performing parameters for optimal results.

5**. \*Classification and Prediction\***

**\*Map: \*** Use the trained DBN to classify new data segments.

**\*Reduce: \*** Compile the predictions into final results.

**6. \*Model Evaluation\***

- \***Map: \*** Measure performance metrics (like accuracy) for each data piece.

- \***Reduce: \*** Combine these measurements to get overall model performance.

**RESOURCE SELECTION:**

1. **Big Data Processing Framework**: Apache Hadoop:

\*Apache Hadoop\* is a widely used framework for distributed storage and processing of large datasets using the MapReduce programming model. It provides:

- \***Hadoop Distributed File System (HDFS)\*:** Stores data across a cluster of machines.

- \*MapReduce\*: Enables parallel processing of data across nodes in the cluster.

2**. Machine Learning Framework**: TensorFlow with Apache Spark

\*TensorFlow\* is an open-source deep learning library developed by Google. When combined with \*Apache Spark\*, it facilitates scalable and efficient distributed training of deep learning models. This setup allows:

3**. Feature Selection:** Scikit-learn with Hadoop Streaming\*Scikit-learn\* is a machine learning library in Python that provides various feature selection algorithms. To integrate it with Hadoop for feature selection:

**4. Hyper-Parameter Tuning:**

\*GridSearchCV\* from \*scikit-learn\* allows systematic tuning of hyper-parameters using cross-validation. For integrating it with TensorFlow:

**5. Integration and Execution**

- \*Apache Hadoop\*: Manage data storage and distributed computing.

- \*Apache Spark\*: Execute data preprocessing, feature selection, and training tasks efficiently across the cluster.

**PROGRAM:** import TensorFlow as tf

from TensorFlow. tetralayers import Dense, Input

from TensorFlow. metamodels import Model

from sklearn. model\_selection import Parameter Grid

# Load selected features data

import pandas as pd

data = pd. read\_csv('/output/selected\_features')

# Assuming the last column is the label

X = datafile;

# Define DBN model

def create\_dbn (hidden\_layers, input\_shape):

inputs = Input(shape=(input\_shape,))

x = inputs

for units in hidden\_layers:

x = Dense (units, activation='rely')

outputs = Dense (1, activation='sigmoid')

model = Model (inputs, outputs)

return model

# Hyper-parameter grid

Param grid = {

'Hidden\_layers': [[64, 32], [128, 64], [256, 128, 64]],

'Batch size': [32, 64],

'epochs': [10, 20]

}

# Grid search for best hyper-parameters

best\_score = 0

best\_params = None

for params in Parameter Grid (Param grid):

model = create\_dbn(params['hidden\_layers'], Shape [1])

model. Compile (optimizer='Adam', loss='binary\_crossentropy', metrics=['accuracy']

model. Fit (X, y, batch size=params ['batch size'], epochs=params['epochs'], validation split=0.2, verbose=0

loss, accuracy = model. Evaluate (X, y, verbose=0)

if accuracy > best\_score:

best\_score = accuracy

best\_params = params

print (fest Hyper-Parameters: {pentagrams}")

print (fest Accuracy: {best\_score}")

**IMPLEMENTATION**

**Connecting the components in cloud:**

**Step-by-Step Implementation in the Cloud**

**# Prerequisites:**

**- \*Cloud Provider Account: \* A**n active account on a cloud provider like AWS, Google Cloud, or Azure.

**- \*Hadoop and Spark Setup: \*** Familiarity with Hadoop and Spark, with an understanding of their setup on cloud platform**s.**

**- \*TensorFlow or Porch Knowledge:** \* Basic knowledge of deep learning frameworks**.**

**BID DATA DEVELOPMENT:**

**Key Areas in Big Data Development**

**1. \*Understanding Big Data: \***

**- \*Definition: \*** Big data refers to datasets that are too large or complex to be effectively managed with traditional data-processing tools.

- \***Characteristics: \*** Often described by the "4 Vs" – Volume, Velocity, Variety, and Veracity.

- \*Use Cases: \* Applications include data analytics, machine learning, predictive analytics, and real-time processing across industries like finance, healthcare, retail, and more.

2. \***Data Infrastructure: \***

- **\*Data Storage**: \* Solutions like Hadoop Distributed File System (HDFS), Amazon S3, and Google Cloud Storage.

- **\*Databases: \*** NoSQL databases (e.g., MongoDB, Cassandra), SQL databases (e.g., MySQL, PostgreSQL), and NewSQL databases.

- \*Data Warehouses: \* Platforms like Amazon Redshift, Google BigQuery, and Snowflake for storing and querying large datasets.

**3. \*Data Processing Frameworks: \***

- \***Batch Processing**: \* Tools like Apache Hadoop and Apache Spark, which handle large-scale data processing in batches.

- \***Stream Processing: \*** Tools like Apache Kafka, Apache Flink, and Apache Storm for real-time data processing.

**4. \*Data Integration and ETL (Extract, Transform, Load): \***

- **\*ETL Tools: \*** Apache NiFi, Talend, Informatica, and Microsoft Azure Data Factory to integrate and transform data from multiple sources.

- **\*Data Pipelines : \*** Building automated pipelines using tools like Apache Airflow and Luigi

**5. - \*Open** **Big Data Tools and Technologies**

**\*Source Tools: \*** Hadoop ecosystem (e.g., Hive, Pig), Spark, Kafka, and Flink.

- \***Cloud Services: \*** AWS Big Data services, Google Cloud Big Data, and Azure HDInsight.

- \***Data Engineering Platforms**: \* Databricks**, Cloudera, and Hortonworks.**

**PERFORMANCE EVALUTION:**

**4. Performance Evaluation: \***

- Assess the performance of the proposed model in terms of accuracy, processing time, scalability, and robustness.

- Compare the proposed model against traditional classification methods and other state-of-the-art models to validate its effectiveness.

**Scope and Deliverables:**

**- \*Data Preprocessing and Analysis**: \*

- Collect and preprocess a large dataset suitable for classification.

- Analyse the data to understand its structure and identify the key features for classification.

- \***MapReduce Implementation: \***

- Design and implement the MapReduce framework for data processing and feature extraction.

- Ensure the MapReduce tasks are optimized for performance and scalability.

- \***Feature Selection:** \*

- Apply feature subset selection techniques to identify the most important features.

- Document the impact of feature selection on model performance and data processing efficiency.

- **\*DBN Development and Tuning: \***

- Build the DBN architecture and conduct extensive hyper-parameter tuning.

- Experiment with various configurations to determine the optimal parameters for the DBN.

- \***Model Evaluation and Comparison:** \*

- Conduct a comprehensive evaluation of the model's classification performance.

- Benchmark the model against other classification methods to highlight its advantages and limitations.

**CONCLUSION:**

**This capstone project successfully developed a scalable classification model for big data by integrating the MapReduce framework, feature subset selection, and an optimized deep belief network (DBN).**

**Key Achievements**:

1**. \*Efficient Data Processing with MapReduce: \***

- MapReduce enabled scalable and efficient handling of large datasets through distributed processing, significantly reducing computation time.

2**. \*Enhanced Model Performance through Feature Selection: \***

- Feature subset selection improved classification accuracy and reduced model complexity by focusing on the most relevant features.

3. **\*Optimized Deep Belief Network: \***

- Hyper-parameter tuning of the DBN resulted in superior classification performance, demonstrating the model's ability to learn complex data patterns effectively.

4. \***Robust Performance Evaluation: \***

- The proposed model outperformed traditional and contemporary classifiers in both accuracy and processing efficiency.

**Future Work:**

- Explore other distributed computing frameworks like Apache Spark.

- Investigate advanced feature engineering techniques.

- Extend to other deep learning architectures such as CNNs and RNNs.

- Test the model in real-world scenarios for practical validation

THANK YOU.