Submitted By : Pranav Agrawal (G01394901)

CS 657

Assignment 2
Report

Contents:

Objective: To undertake comprehensive data preparation including label encoding, extraneous data removal, addressing missing values, and text normalization. Due to dataset asymmetry, the report aims to implement undersampling techniques to ensure reliable model training. The core objective is to train and validate four machine learning models: Logistic Regression, LinearSVC, RandomForestClassifier, and MultilayerPerceptronClassifier, leveraging feature extraction and transformation strategies. The report will further emphasize optimizing these models through cross-validation, meticulous parameter tuning, and evaluating their performance based on metrics such as accuracy and F1 scores.

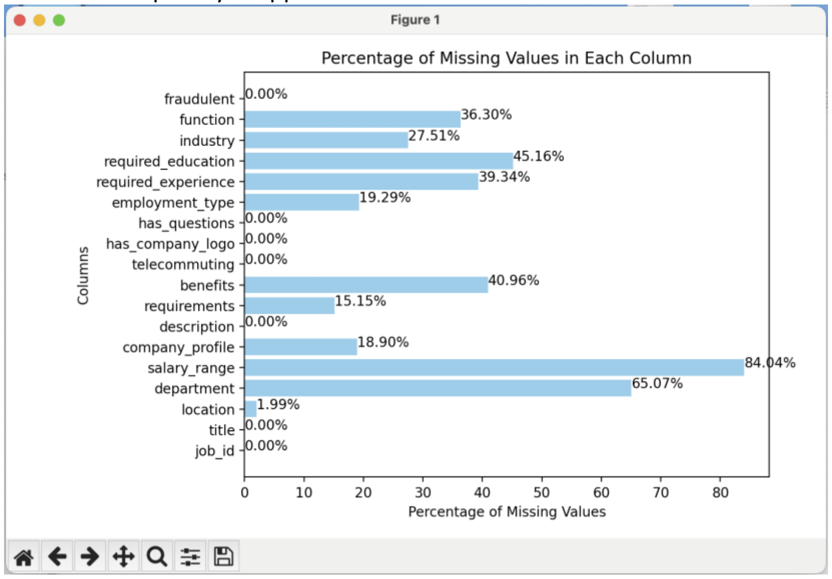
Data Description: The dataset under study comprises 18,000 job descriptions, of which approximately 800 have been identified as fraudulent. These descriptions encompass not only the textual details of the job but also meta-information related to them. The intrinsic nature of this dataset makes it a valuable resource for the development of classification models. The primary aim is to utilize this data to train models that can effectively discern and flag fraudulent job descriptions based on the patterns and characteristics they exhibit.

Pseudo-code:

```
1.Data Cleaning:
  a. Filter out invalid rows based on 'fraudulent' column values
  b. Calculate and record missing value counts and percentages per column
  c. Drop columns with more than a certain percentage of missing values
  d. Clean textual data (remove non-alphanumeric characters, convert to lowercase, etc.)

2. Class Balancing:
  a. Separate data into majority and minority classes based on 'fraudulent' column
  b. Calculate the appropriate sampling fraction for the majority class
  c. Undersample the majority class to balance with the minority class
  d. Merge the undersampled majority class with the minority class

3. Text Data Preprocessing and Feature Extraction:
  a. Tokenize the text data into words
  b. Remove stopwords from the tokenized words
  c. Convert the processed text into feature vectors using HashingTF and IDF methods
  d. Run the text data through a pipeline that includes the above stages

4. Model Training Preparation:
  a. Split the data into training and testing sets
  b. Initialize the machine learning models (LogisticRegression, LinearSVC,     RandomForestClassifier,
MultilayerPerceptronClassifier
MultilayerPerceptronClassifier)

5. Model Tuning and Evaluation:
  a. Define the parameter grids for the models
  b. Setup cross-validation with the models, parameter grids, and evaluation criteria
  c. Fit the models on the training data and perform cross-validation
  d. Evaluate each model based on testing data, recording performance metrics (accuracy, F1 score, etc.)
  e. Identify and record the best performing parameters
```

Data Preprocessing: Following are the steps performed for data preprocessing:

- **Label Validation:** Any rows containing fraudulent values other than 0 (non-fraudulent) or 1 (fraudulent) were removed to ensure data consistency.
- **Handling Missing Values:** The number of missing values for each attribute was computed. This count was converted to percentages to understand the magnitude of missing data for each attribute. Columns having more than 1% of missing values were identified and subsequently dropped from the dataset.



- **Text Cleaning:** Non-alphanumeric characters were removed. Consecutive spaces were reduced to single spaces. All texts were converted to lowercase.
- **Balancing the Dataset:** Given that the dataset was highly imbalanced. The data was separated into two classes: majority (non-fraudulent) and minority (fraudulent). The majority class was under sampled using the sample By method in PySpark, ensuring the majority class had approximately the same number of rows as the minority class. The under sampled majority class and the minority class were then combined to produce a balanced dataset.
- **Text Vectorization:** Job descriptions were tokenized, splitting the text into individual words. Stopwords (common words that typically don't carry meaningful information in text analysis) were removed. The resulting words were transformed into vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. This method quantifies the importance of each word in a document relative to its frequency in the entire corpus. HashingTF was used to convert the words into raw feature vectors and reduced number of features to 21000 using numFeatures parameter, and then the IDF method was applied to convert these raw features into TF-IDF features.
- **Data Splitting:** The pre-processed data was then split into training and test sets in a 70:30 ratio.

Train, Test and validation:

to train, validate, and test the following models are used:

- LogisticRegression
- LinearSVC
- RandomForestClassifier
- MultiLayerPerceptronClassifier

LogisticRegression

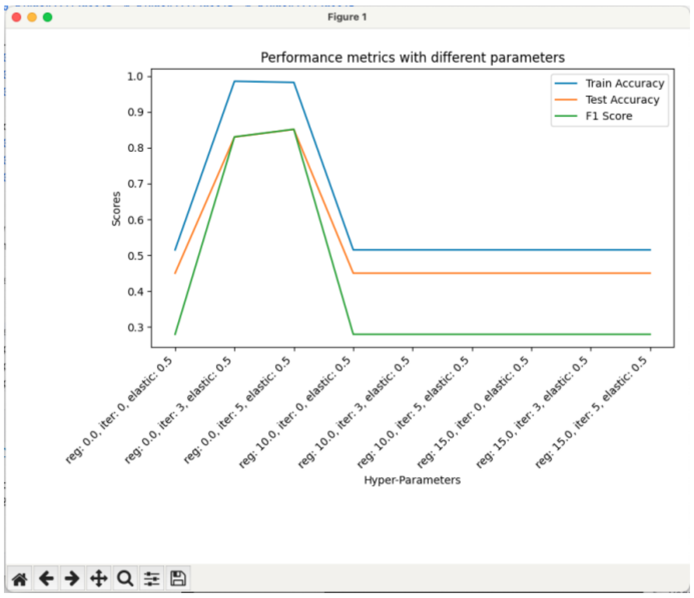
In PySpark, LogisticRegression from the 'pyspark.ml.classification import LogisticRegression' library is used for binary classification or multinomial logistic regression. Employing the DataFrame API, it utilizes features columns for training, supports hyperparameter tuning, and offers robustness through regularization, elastic net mixing, and handling sparse data efficiently.

Tuning Hyperparameters: I altered regParam, maxIter, elasticNetParam to get following scores :

S.No	regParam	maxIter	elasticNetParam	Train Accuracy	Test Accuracy	F1 Score
1	0.0	0	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716

2	0.0	3	0.5	0.9858044164037855	0.8301158301158301	0.8305351027021483
3	0.0	5	0.5	0.9826498422712934	0.8513513513513513	0.8517132947401906
4	10.0	0	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716
5	10.0	3	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716
6	10.0	5	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716
7	15.0	0	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716
8	15.0	3	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716
9	15.0	5	0.5	0.5149842271293376	0.4498069498069498	0.27910790760324716

Graph:



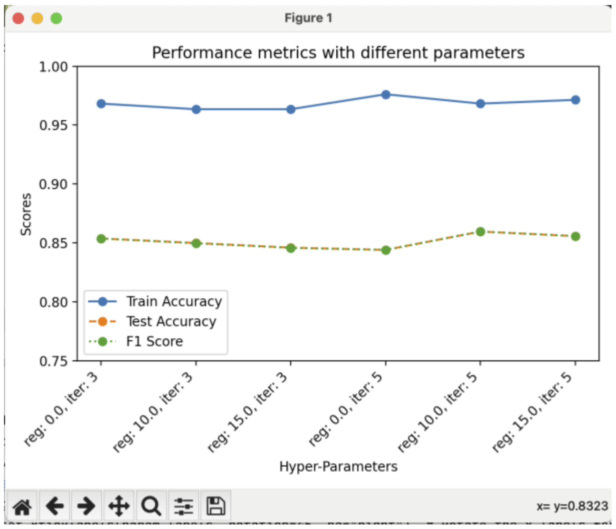
So I got best result at regParam=0, maxIter=5, elasticNetParam=0.5, because here training, testing and F1 score is highest.

LinearSVC
PySpark's LinearSVC, part of the 'pyspark.ml.classification import LinearSVC' suite, is a classifier implementing linear Support Vector Machines (SVM). It's designed for binary classification tasks, using a linear kernel to separate data points with a hyperplane. With features like regularization parameter tuning and threshold adjustments, LinearSVC supports large-scale, high-dimensional datasets, offering efficient handling and parallel processing capabilities inherent to Spark's environment. This classifier is particularly suited for scenarios where precision is critical, as it focuses on maximizing the margin between different classes, albeit at the computational cost typical for SVMs.

Tuning Hyperparameters: I altered regParam, maxIter to get following result :

S.No	regParam	maxIter	Train Accuracy	Test Accuracy	F1 Score
1	0	3	0.9682539682539683	0.8538011695906432	0.8537545020559605
2	10	3	0.9634920634920635	0.8499025341130604	0.8496548193279116
3	15	3	0.9634920634920635	0.8460038986354775	0.845800151298628
4	0	5	0.9761904761904762	0.8440545808966862	0.8440545808966862
5	10	5	0.9682539682539683	0.8596491228070176	0.8596811244967068
6	15	5	0.9714285714285714	0.8557504873294347	0.8557790081456281

Graph:



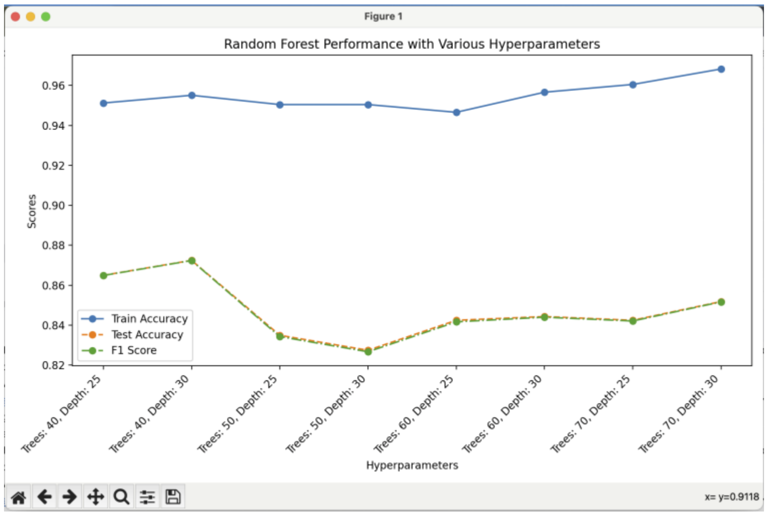
So I got best result at regParam=10, maxIter=5, because here testing and F1 score is highest and further on increasing the value, overfitting is evident.

RandomForestClassifier
The RandomForestClassifier in PySpark, part of pyspark.ml.classification, utilizes ensemble learning with multiple decision trees to ensure robust, accurate predictions while preventing overfitting. Ideal for large, complex datasets, it offers insights on feature importance, customizability in modeling, and efficient parallel processing, capitalizing on Spark's distributed computing features.

Tuning Hyperparameters: I altered numTrees, maxDepth to get following result :

S.No	numTrees	maxDepth	Train Accuracy	Test Accuracy	F1 Score
1	40	25	0.9512006196746708	0.8649155722326454	0.8648299143225262
2	40	30	0.9550735863671572	0.8724202626641651	0.8723132719048772
3	50	25	0.9504260263361735	0.8348968105065666	0.8342430844324369
4	50	30	0.9504260263361735	0.8273921200750469	0.8266240785040735
5	60	25	0.9465530596436871	0.8424015009380863	0.8416184388338037
6	60	30	0.9566227730441518	0.8442776735459663	0.8439491002228672
7	70	25	0.9604957397366383	0.8424015009380863	0.842041097820525
8	70	30	0.9682416731216111	0.851782363977486	0.8516047258024152

Graph:



Hence I got best result at numTrees=40, maxDepth =30, on the basis of test accuracy and training accuracy, and also got highest F1 Score at this point.

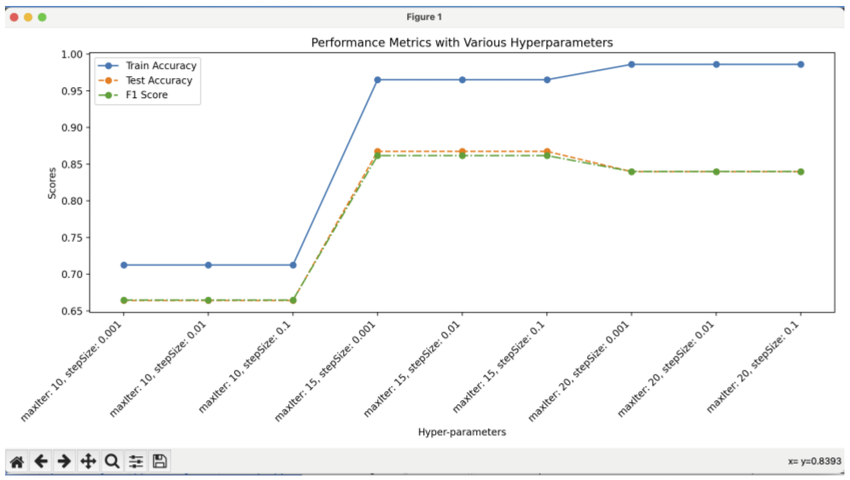
MultilayerPerceptronClassifier

The MultilayerPerceptronClassifier in PySpark is a neural network model for classification. It's initialized with parameters defining input ('features'), output ('fraudulent'), and network architecture (layers=[21000, 24, 2] indicating input, hidden, and output layers). The model learns during training and then predicts classes for new data, with its performance depending on architecture and data quality.

Tuning Hyperparameters: I altered maxIter, stepSize to get following result :

S.No.	maxIter	stepSize	Train Accuracy	Test Accuracy	F1 Score
1	10	0.001	0.7125097125097125	0.6640926640926641	0.6646938853665679
2	10	0.01	0.7125097125097125	0.6640926640926641	0.6646938853665679
3	10	0.1	0.7125097125097125	0.6640926640926641	0.6646938853665679
4	15	0.001	0.9650793650793651	0.8674463937621832	0.8616331549409019
5	15	0.01	0.9650793650793651	0.8674463937621832	0.8616331549409019
6	15	0.1	0.9650793650793651	0.8674463937621832	0.8616331549409019
7	20	0.001	0.986013986013986	0.8397683397683398	0.8400010432208574
8	20	0.01	0.986013986013986	0.8397683397683398	0.8400010432208574
9	20	0.1	0.986013986013986	0.8397683397683398	0.8400010432208574

Graph:



This is observed that best test accuracy and F1 score can be found at maxIter = 15, however stepSize parameter doesn't have any impact on any score, and on increasing the value of maxIter further, overfitting is evident.

Conclusion:
I have successfully completed the careful data preparation process. Label encoding, unnecessary information removal, missing value handling, and significant text normalisation were all performed. Recognizing the dataset's inherent asymmetry, a strategic decision was made to adopt under sampling, ensuring the data's suitability for dependable model training.

With the well preprocessed data, I have proceeded with the training and validation of the proposed machine learning models: Logistic Regression, LinearSVC, RandomForestClassifier, and MultilayerPerceptronClassifier, with the foundation created through balanced dataset. These models, which will make use of advanced feature extraction and transformation algorithms, will be rigorously optimised. The following steps will prioritise fine-tuning by cross-validation, precise parameter modifications, and a thorough evaluation based on key performance metrics such as accuracy and F1 scores.