Data Mining

Homework 2

Due: 17/11/2024, 23:59

Instructions

You must hand in the homeworks electronically and before the due date and time.

The first homework has to be done by each **person individually**.

Handing in: You must hand in the homeworks by the due date and time by an email to Gianluca (`decarlo@diag.uniroma1.it`) that will contain as attachment **(not links to some file-uploading server!)** a .zip file with your answers. The filename of the attachment should be `DM_Homework_2__StudentID_StudentName_StudentLastname.zip`;
for example:
`DM_Homework_2__1235711_Robert_Anthony_De_Niro.zip`.
The email subject should be
`[Data Mining] Homework_2 StudentID StudentName StudentLastname`;
For example:
`[Data Mining] Homework_2 1235711 Robert Anthony De Niro`.
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after the deadline then contact Gianluca.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

For information about collaboration, and about being late check the web page.

Problem 1. In this exercise we will get some practice in using some Python libraries, for downloading web pages, parsing them, and performing some analysis. We will obtain data about articles available on Amazon. For downloading the web pages you may use the package `Requests` or the package `urllib2`. To parse the page you can either use regular expressions through the package `re` (it is anyway a good idea become familiar with regular expressions), or, probably better, use an HTML/XML parser. The `Beautiful Soup` package is a good one but it loads the whole file in memory. This is fine for this problem, since the pages to parse are small, but be careful if you want to use it on large XML files; for those ones check the `lxml` library and the tutorial at `http://www.ibm.com/developerworks/xml/library/x-hiperfparse/`.

Write a program that will download from `https://www.amazon.it/` and parse all the products output for a given keywork. You can search for a product page using the url `https://www.amazon.it/s?k=KEYWORK&page=X`, where KEYWORD is the product you want to look for and X is the page number. For this exercise use the keywork *computer* (you will see that you'll have many different products, like PCs, notebook covers, stands, etc.). Save the products in a tab-separated value (TSV) file in which, for any product (one line per product) you have: *product description*, *price*, whether it is a *prime product or not*, *url* of the product page, *number of stars* of the product. Because you will make a lot of calls to the Amazon site, make sure that you have a delay (use: `sys.sleep()`) between different downloads of Amazon pages, to avoid being blocked. After downloading, do the following:

1. For each product description, use `NLTK` Python library to perform any preprocessing you may find useful (stemming, normalization, stopwords removal, etc).

2. The next step is to build a search-engine index. First, you need to build an inverted index, and store it in a file. Build an index that allows to perform proximity queries using the cosine-similarity measure. Then build also a query-processing part, which given some terms it will bring the most related product.

Hand in the code, along with some examples of queries and screenshots of the results. Try both short and long queries (like a full product description).

Problem 2. Implement Problem 1 using Apache Spark (with the already downloaded pages).

Problem 3. In this homework we will use PySpark's MLlib to build, tune, and evaluate machine learning models that predict flight delays based on historical data.

Dataset

Use the Flight Delay and Cancellation Dataset (2019-2023) from Kaggle, which contains comprehensive flight information, including delays and cancellations from 2019 to 2023.

Data Preparation

Import the `flights_sample_3m.csv` dataset into a PySpark DataFrame. Examine the schema and data types, checking for null or missing values, and identify any anomalies or outliers. Perform basic exploratory data analysis (EDA) to understand distributions and relationships within the data. Handle missing values appropriately, either by dropping or imputing them.

Identify features relevant to predicting flight delays, such as departure time, airline, origin, destination, and weather conditions. Clean the data by removing irrelevant features and inconsistencies. Convert categorical variables (e.g., airline codes, airport codes) into numerical format using `StringIndexer` and `OneHotEncoder`. Assemble all feature columns into a single feature vector with `VectorAssembler`. Split the dataset into training and testing sets (e.g., 80% training, 20% testing).

Model Development

Frame the problem as a binary classification task, predicting whether a flight is delayed by more than 15 minutes. Develop two machine learning models using PySpark's MLlib: Logistic Regression and Random Forest Classifier.

Enhance model performance through hyperparameter tuning. To define hyperparameter grids for each algorithm use `ParamGridBuilder` (`regParam`, `elasticNetParam` for Logistic Regression, `numTrees`, `maxDepth` for Random Forest). Implement five-fold cross-validation with `CrossValidator` and use `BinaryClassificationEvaluator` to evaluate models based on metrics like area under the ROC curve.

Evaluation and Reporting

Apply the best models from cross-validation to the test set to make predictions. Evaluate each model's performance by computing metrics such as accuracy, precision, recall, F1-score, and AUC. Use `MulticlassClassificationEvaluator` or `BinaryClassificationEvaluator` as appropriate. Construct and interpret confusion matrices to understand the models' error types.

Visualize results to aid interpretation. Create plots during EDA to illustrate data distributions and correlations. Plot ROC curves for both models to compare performance visually, and generate bar charts to display feature importances from the Random Forest model.

Document your process thoroughly, commenting code for clarity. If using a Jupyter notebook, include markdown cells explaining each step and your reasoning. Compose a concise report summarizing your methodology, results, and conclusions. Discuss any challenges faced and how you addressed them, insights into model performance, potential limitations, and suggestions for future improvements.

Deliverables

1. A well-documented PySpark script or Jupyter notebook covering your workflow from data loading to model evaluation.
2. A written report (PDF or Markdown) detailing your approach, findings, and interpretations, including visualizations.
3. Relevant plots and graphs included in your report to support your analysis.

Optional Extensions

For further exploration, consider implementing additional classifiers like Gradient-Boosted Trees or Neural Networks to compare performance with the initial models.