****

**Assignment 5 -** Method (Algorithm Overview and Implementation in Zeppelin)

**Vehicle Traffic, Provided by City of Aarhus in Denmark**

**Presented by:** Amit Pandey

**Guided by:** Professor Sylvain Jaume

**Course:** DS670 - Capstone

**INDEX**

* **Introduction and Data Structure**
* **Methods applied**
* **Evaluation/Building the model**
* **Prediction**
* **Conclusion/summary**

**Introduction:**

The urban traffic congestion is transforming into an epidemic all over the world. The brisk increase in vehicle traffic has become one of the critical problems faced by cities all over the world. As a result of the constant traffic congestion, transportation cost has increased significantly due to all the time wasted on the road and the corresponding fuel cost. This proposal looks at how the Vehicle Traffic data collected can be used in making Aarhus a smart city in terms of traffic guidance and management.

**Data description/structure:**

A collection of datasets of vehicle traffic in the city called Aarhus in Denmark, observed between two points for a set duration of time over a period certain months (449 observation points in total). The data is available in raw (CSV) and semantically annotated format using the Citypulse information model. There are total four (4) different datasets over different durations available:

1st data set: February 2014 to June 2014

2nd data set: August 2014 to September 2014

3rd data set: October 2014 to November 2014

4th data set: July 2015 to October 2015

**Data processing:**

The data is available in raw (CSV) and semantically annotated format using the citypulse information model. For this project, I’m going to use the “.csv” format only as it’s easy to manipulate and analyze the data. The data analysis will be done in Apache Zeppelin environment that is being taught in the class. To test out the data, I took the first dataset from February 2014 to June 2014 and saved all the files in a folder “traffic\_feb\_june” in my Downloads directory. Using Zeppelin Spark, I was able to consolidate all the files in one table called roadtraffic. Using SQL table, I ran a few queries to look into the data set and get a summary. I have only looked at the first dataset (Feb 2014 – June 2014) for now to do the analysis.

**Method:** As explained in Figure 1, I plan to use the following method to collect, process and predict the datasets.

**Figure 1 – Method**

**Pseudo code:**

1. **Collect and download the datasets into one folder.**

Based on the data provided for Aarhus, we will download and collect all the datasets in one folder. As the data is available in the raw (CSV) and semantically annotated format using the Citypulse information model, we will use the raw (CSV) format to process and analyze the ata.

1. **Load data and combine all data sets in Zeppelin.**

Using Zippelin, we will load and combine all the files (449) in the datasets to analyze the information.

1. **Process datasets and label combined data based on velocity.**

Depending on the programming language (R, Python or SQL), we decide to use on Zeppelin, we may need to import some libraries to do the scientific calculations. We may need the libraries like “NumPy” in Python or “Caret” (short for Classification And Regression Training) in R to attempt to streamline the process for creating predictive models. These libraries contains tools for data splitting, pre-processing, feature selection, model tuning using re-sampling, variable importance estimation, .as well as other functionality.

We’ll also do an analysis to see if the data needs to be cleaned and if there are any errors that may corrupt our analysis. We also want to make sure that the null values are set to zero and take care of missing values. We will check for common errors like: missing values, corrupted values, data range errors, etc. We will have to look through file rows and columns and sample test values to see if the values make reasonable sense.

1. **Split Datasets into Training (80%) and Testing Data (20%).**

As we are going to do some prediction, it would be best to split our combined dataset into two different datasets: Training (80%) and Testing (20%). The Training dataset will be used to build and test different statistical mode and the Testing dataset will be used to evaluate (cross-validate) our model and assumption. We do the splitting and create the Test partition to provide us with a fair assessment of our prediction model build.

There is no single method for selecting the extent of training and testing data. Some people use 90/10 and some prefer 80/20. In any case, doing as such can cause bias the classification results. For selecting the right split, we can use either “N-Fold cross validation” or “K-fold cross validation”. In N-Fold cross validation, we randomize the data and create N equal partition and choose the Nth partition for testing dataset and N-1 partition for the training. In the training set, we can apply “K-fold cross validation” to validate and find the best parameter. This will take out any bias out of our assumptions.

1. **Apply machine learning algorithm and fit models based on the algorithm.**

Next we’ll have to take a look to employ an algorithm model that will best fit the relation among the attributes and class label the data. The model created by the algorithm must fit both the training data and also accurately predict the test data.

**Figure 2:** Model for building a classification model.

We will use different classification algorithms to classify the data such as:

**KNN:**

K closest neighbors (KNN) is a basic algorithm that stores all accessible cases and characterizes new cases in view of a closeness measure. KNN has been utilized as a part of measurable estimation and example acknowledgment as of now in the start of 1970 as a non-parametric strategy. KNN is a non parametric lazy learning calculation. When we say a strategy is non parametric, it implies that it doesn't make any assumption on the data distribution. This is quite helpful, as in practical world, the greater part of the practical information does not comply with the ordinary hypothetical suppositions made.

It is additionally a lazy algorithm which means it doesn't utilize the training data to do generalize. Basically, there is very little training phase. This implies the training data is really quick. Absence of speculation implies that KNN keeps all the training information. All the more precisely, all the training information is required amid the testing stage. This is different than algorithm techniques like SVM where you can discard of all non-support vectors with no issue.

**Random forest decision tree:**

Random forests or random decision forests are a group learning technique for classification, relapse and different undertakings, that work by developing a huge number of decision trees at preparing time and yielding the class that is the method of the classes (classification) or mean forecast (relapse) of the individual trees. Random decision forests adjust for decision trees' propensity for over fitting to their training set. Forests develops numerous classification trees. To classify another question from an information vector, put the info vector down each of the trees in the timberland. Each tree gives a classification, and we say the tree "votes" for that class. The backwoods picks the classification having the most votes (over every one of the trees in the timberland).

**Support Vector Machines (SVM) model:**

SVMs also support vector networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Margin/support to make a group for the training data. A SVM is a discriminative classifier also defined by a separating hyper plane. In other words, in supervised learning, given labeled training data, the algorithm outputs an optimal hyper plane that can categorize new examples.

N**eural Networks:**

Neural networks, sometime also suggested as connectionist systems, are a computational approach, which relies on upon a broad gathering of neural units or fabricated neurons, openly exhibiting the way a natural cerebrum deals with issues with far reaching clusters of normal neurons related by axons. Each neural unit is related with various others, and associations can actualize or inhibitory in their effect on the incitation state of related neural units. Each individual neural unit may have a summation limit which merges the estimations of each one of its wellsprings of information together. There may be an edge limit or confining limit on each affiliation and on the unit itself: to such a degree, to the point that the banner must beat the most distant indicate before causing diverse neurons. These structures are self-learning and arranged, rather than unequivocally altered, and surpass desires in domains where the plan or highlight acknowledgment is difficult to express in a standard PC program.

1. **Evaluate the models on the Testing Data: Cross validate**

Once we’ve build the algorithm models, we would need to evaluate the model on the testing data. The evaluation of the performance of the classification model we will be based on the number of records correctly and incorrectly assessed by the model, which will be done through confusion matrix.

We will do a cross validation to check the accuracy of the training model. Cross-validation, sometimes called rotation estimation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. The aim of the cross validation is also to avoid over-fitting of the model.

A confusion matrix is a table that is frequently used to depict the execution of a classification model (or "classifier") on an arrangement of test information for which the genuine qualities are known. The confusion matrix itself is generally easy to see, however the related phrasing can be confusing. Here’s an example of confusion matrix:



Here are the basic terms used in the confusion matrix:

* **True positives (TP):** These are cases in which we predicted yes (for example, when they have the disease), and they do have the disease.
* **True negatives (TN):** We predicted no, and they don't have the disease.
* **False positives (FP**): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
* **False negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Even though a confusion matrix will provide the details needed to figure out how well the classification model works, we can figure out the summary of the information with a single number to make it convenient to compare the performance of different models, through the following measures:

* **Accuracy:**  Number of correct prediction/total number of predictions to determine the correctness of the classifier. It can be calculated as:
  + (TP+TN)/total = Accuracy
* **Misclassification Rate:** To determine how often the model is wrong:
  + (FP+FN)/total = Misclassification rate.
  + Also, can be determined by 1 minus Accuracy
  + It is also called the "Error Rate"
* **Precision:** When the model predicted yet, how often is that correct.
  + TP/predicted yes = Precision
* **Recall/True Positive Rate:** When the model actually says “yes”, how often does it predict yes?
  + TP/actual yes = True Positive Rate:
  + This is also known as the "Sensitivity" or "Recall"
* **False Positive Rate:** When the model returns no, how often does it predict “yes”.
  + FP/actual no = False Positive Rate
* **Specificity:** When it's actually no, how often does it predict no.
  + TN/actual no = Specificity
  + Also, equals 1 minus False Positive Rate
* **Prevalence:** How often does the yes condition actually occur in our sample.
  + actual yes/total = Prevalence.

The accuracy of the model can also be calculated through a few other measures like:

* **Null Error Rate:** Null error rate is how often we would be wrong if we always predicted the majority class. This can be a useful baseline metric to compare our classifier against.
* **F Score:**  F-score is the weighted average of the true positive rate (recall) and precision.
* **ROC Curve:** This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class.

1. **Make a conclusion/prediction.**

Based on the confusion matrix and the evaluation measurers (as defined above), we are going to make a model prediction on the accuracy of the model.