University of Waterloo

ECE 657A:
Data and Knowledge Modeling and Analysis

Spring 2021

Assignment 2:

Classification using Naive Bayes, decision tree,
random forest, XGBoost

Due: July 11, 2021 at 11:59pm EST

Overview

Collaboration: You may do your work individually or in pairs. You may collaborate with other students in the class on the right tools to use and setting up your programming environment but work on your own solution but be done by members of your group alone. Both members of the pair must sign up for a PairGroup in LEARN *and* in Crowdmark.

Hand in: One report per person or pair, via the CROWDMARK website in PDF, or image, format. You will need to divide the PDF up into one file for each `[CM#]` question. These files can be multiple pages. Some "questions" in this assignment have no output so no pdf if needed. It is best to start each of `[CM#]` solution on a new page and then drag and drop each onto the relevant question in Crowdmark. You should receive an invite to Crowdmark by email. Overlap and duplicated text between questions is alright, as long as the *entire answer* for each question fully contained within that question's pdf file. Also submit the code/scripts needed to reproduce your work as a python jupyter notebook to the LEARN dropbox.

Specific objectives:

- Load the dataset and perform some data cleaning and exploratory plots to understand the dataset.
- Study how to apply some of the methods discussed in class and gain experience on the use of classification algorithms: Naive Bayes, decision tree, random forests, XGBoost.

Tools: You can use libraries available in python. You need to mention which libraries you are using, any blogs or papers you used to figure out how to carry out your calculations.

Dataset - DKMA-Covid19-Jan-States

This dataset uses statistics on COVID-19 cases in US states in January of 2021. The original data comes from the following two sources:

- John Hopkins University CSSE COVID-19: `https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data`
- US 2020 Census

The datapoints describe various information about the population and Covid pandemic situation in **50 US States** and over each day of the month of **January 2021**. Each record gives Covid information for that *day* and demographic data for the *whole state* as well. Note the demographic data is duplicated for each state since for all days since it doesn't change that quickly and is based on data from the last census. Also note that the data was combined together from multiple sources and not processed for uniform number formats and was not normalized. As far as we are aware there is no missing data, but there are certainly some states which more extreme values than others, so consideration of outliers is reasonable. All of this will be initial pre-processing you will need to carry out.

File Descriptions:

- `dkmacovid_train.csv` - main training data with input features and three binary labels.
- `dkmacovid_kaggletest_features.csv` - feature for a test set, with "Id" column for Kaggle competition.
- `dkmacovid_kaggletest_labels.csv` - labels for the kaggle test set...oh, you don't get that one :)

Fields Descriptions:

- **Day**: Date in January 2021 ranging from Jan 2 to Jan 31.
- **State ID**: Arbitrary ID number for each state, based on alphabetical order. Note there are 51 states since the District of Columbia is also included.
- **State**: Name of the US State.
- **Lat**: Latitude for the geographic centre of the state.
- **Long_**: Longitude for the geographic centre of the state.
- **Active**: Number of active, tracked COVID-19 cases that day in that state.
- **Incident_Rate**: see original data source
- **Total_Test_Results**: see original data source
- **Case_Fatality_Ratio**: see original data source

- **Testing_Rate**: see original data source
- **Resident Population 2021 Census**: see original data source
- **Population Density 2021 Census**: see original data source
- **Density Rank 2021 Census**: see original data source
- **SexRatio**: see original data source

Label Description:

Each row of this data dataset represents a particular **day** in January, 2021 in a particular **US State** and has three binary labels which are:

- `True` if is the corresponding count went **up** from the previous day.
- `False` if is the corresponding count went **down** from the previous day.

The three binary labels are:

- **Confirmed**: Was there a daily increase in the **confirmed total cases** of COVID-19 in that state on that day?
- **Deaths**: Was there a daily increase in the **number of deaths** from COVID-19 in that state on that day?
- **Recovered**: Was there a daily increase in the number **of people recovered** from COVID-19 in that state on that day?

Note that the labels are fairly **unbalanced**, so there are quite a few more `True` cases than there are `False` cases. **Recovered** is the most balanced, so it should be easier to train. **Deaths** is next and **Confirmed** is the least balanced.

1 Data Pre-processing and Preparation

`[CM1]` Report a short summary of the pre-processing steps you applied to the data to make it usable for analysis. This should at a minimum include discussion of outliers and normalization (which features, what kind of normalization, if any, is needed and appropriate?). What do you do with "Day", "State" and "State ID"?

2 Representation Learning

`[CM2]` Apply PCA and LDA onto the dataset, on an appropriate subset of the features. For LDA use each of the labels individually to train it.

- Use a scree-plot to look at the cumulative variance represented by the PCA eigenvectors, give advice on the best number number of reduced features to use to represent the data.

- Plot the datapoints, or a useful subset of them (select out one day per state? or use color by State?) on the first two PCA and LDA feature vectors.
- Does the LDA method provide better results for one label more than the others?

For the remaining questions: you can try using both the **original dataset** and a **new hybrid dataset** which you construct which replaces (some or all) of the numeric value features with a subset of PCA or LDA features. You must report results for both for Random Forests and Gradient Tree Boosting.

3 Decision Trees Classifier

`[CM3]` Classify the data using Decision Trees Tune the hyper-parameters of the classifier using k-fold cross validation and sklearn functions. Evaluate the best value for the number of trees and maximum depth of trees. You can choose k yourself based on need. For decision trees try a range of parameters at least including the following:

- max depth: {3, 5, 10, None}
- *None: (grow until leaf contains 2 datapoints)*

For this, plot the mean accuracy versus the maximum depth. Also, examine the final resulting splitting rules used for the trees. Do they indicate any interesting patterns that explain the data? Use the original feature space only for this, not the extracted features from PCA or LDA.

4 Random Forest Classifier

`[CM4]` Classify the data using Decision Trees Tune the hyper-parameters of the classifier using k-fold cross validation and sklearn functions. Evaluate the best value for the number of trees and maximum depth of trees. You can choose k yourself based on need.

For random forest try a range of parameters at least including the following:

- number of trees: {5, 10, 50, 150, 200}
- max depth: {3, 5, 10, None}

For this, the plot should be a **heat plot**. You should have (5 * 4) mean accuracies for different values of number of trees and maximum depth. Report results using original features and using PCA or LDA features, see Part 2.

5 Gradient Tree Boosting

[CM5] Classify the data using Decision Trees Tune the hyper-parameters of the classifier using k-fold cross validation and sklearn functions. Evaluate the best value for the number of trees and maximum depth of trees. You can choose k yourself based on need.

For Gradient Tree Boosting (on sklearn it is `GradientBoostingClassifier`):

- number of estimators: {5, 10, 50, 150, 200}

For this, plot the mean accuracy versus the number of estimators.
Note: the number of 'trees' grown by GBT is `n_classes` $\times$ `n_estimators` *but this is handled automatically.* You can leave the other parameters as default in sklearn. Report results using original features and using PCA or LDA features, see Part 2.

6 Naive Bayes Classifier

[CM6] Classify the data using the Naive Bayes Classifier. You can use GaussianNB in sklearn. Tune the hyper-parameters of the classifier using 10-fold cross validation and sklearn functions. Use the original feature space only for this, not the extracted features from PCA or LDA.

Evaluate the best value for the var_smoothing among the values {1e-10, 1e-9, 1e-5, 1e-3, 1e-1} and report the results using any performance measures you choose. Can you explain the impact of the smoothing parameter?

7 Interpretability

[CM7] Explain the performance of NB compared to the decision tree approaches in Question 1. Use the original feature space only for this, not the extracted features from PCA or LDA. Can you use the learned parameters of NB to make an interpretation about the data and compare this to the single Decision Tree model?

8 Kaggle

[CM8] Instructions will be posted on LEARN about the related Kaggle competition for this assignment once it is set up. You can use any method from the assignment to produce your answer submission. The description on Kaggle will tell you which label(s) you need to predict and the format. Data from five states *Washington, Tennessee, Iowa, Texas, Illinois* has been held out from the `train` set and will be used for testing your trained algorithms on Kaggle and is available on LEARN (or Kaggle) as `dkmacovidkaggletestfeatures.csv`. In this part of your report on Crowdmark simply enter your group name and url on kaggle, so we can find your results. You can also report your results from

your last submission to kaggle here. (We will see another score using held out data you do not have access to).

Notes

You might find the following links are useful to solve this assignment:

- `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html`