Chronic Absenteeism Rate Prediction (CARP) Architectural Decisions

December 1, 2019

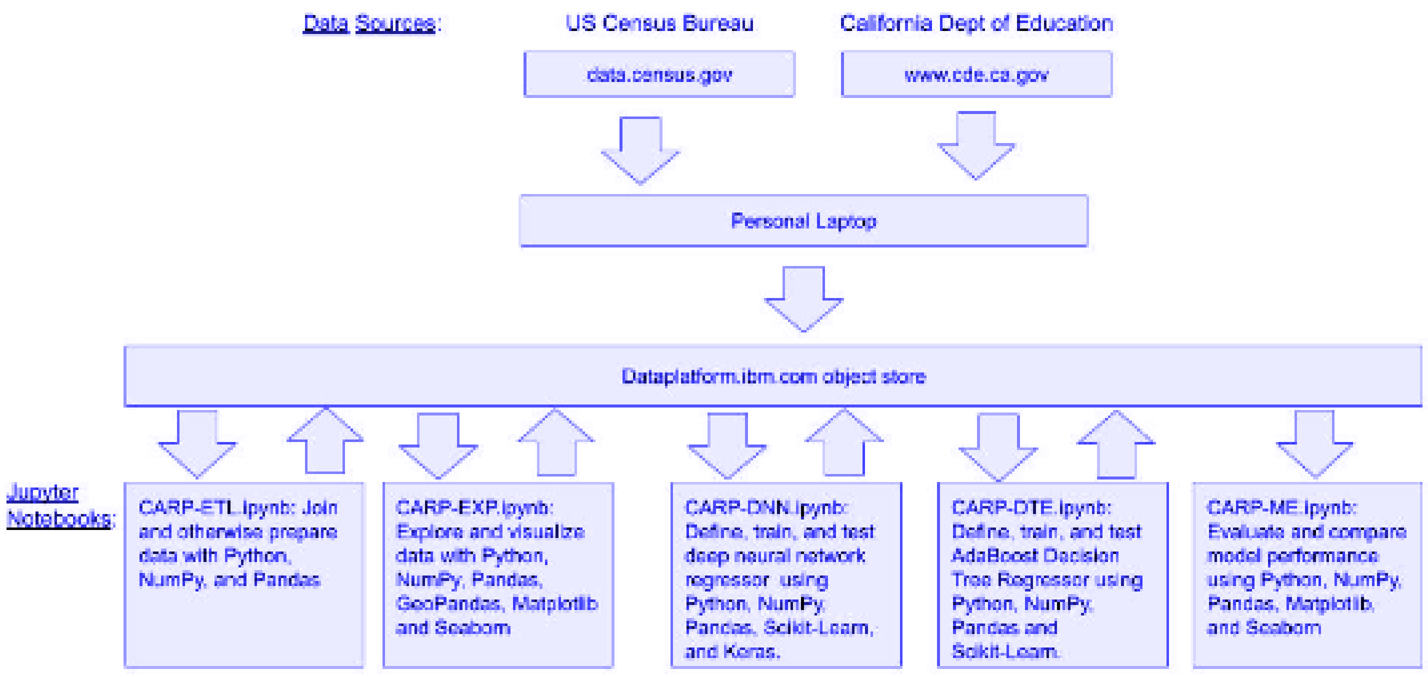
Background and Use Case

Chronic absenteeism occurs when a student in grades K-12 misses 10% or more of the school days in any year for any reason, as defined by the California Department of Education (some other organizations have different thresholds, e.g 15%). It is a strong predictor of low academic achievement, including failure to complete secondary school. CARP is designed for use and enhancement by data scientists supporting educational and social services administrators and policymakers in answering the following questions:

1. What demographic factors influence the rate of chronic absenteeism?
2. What rates of chronic absenteeism can we expect in particular census tracts?

CARP addresses these questions with data from the US Census Bureau and the California Department of Education. It uses data from the 2013-2017 US Census American Community Survey (ACS) to predict 2018 chronic absence rates by US Census tract in Los Angeles (LA) County, California.

Architectural Components Overview


Data Sources:
US Census Bureau
California Dept of Education
data.census.gov
www.cde.ca.gov
Personal Laptop
Dataplatform.ibm.com object store
Jupyter Notebooks:
CARP-ETL.ipynb: Join and otherwise prepare data with Python, NumPy, and Pandas
CARP-EXP.ipynb: Explore and visualize data with Python, NumPy, Pandas, GeoPandas, Matplotlib and Seaborn
CARP-DNN.ipynb: Define, train, and test deep neural network regressor using Python, NumPy, Pandas, Scikit-Learn, and Keras.
CARP-DTE.ipynb: Define, train, and test AdaBoost Decision Tree Regressor using Python, NumPy, Pandas and Scikit-Learn.
CARP-ME.ipynb: Evaluate and compare model performance using Python, NumPy, Pandas, Matplotlib, and Seaborn

Data Subjects and Sources

This product uses ACS data from the US Census Bureau, because it contains data on income, employment, and other subjects that could affect chronic absenteeism, which is defined as missing more than 10% of days in a school year for any reason. I have also chosen data on chronic absenteeism by US Census Tract for LA County, California from the California Department of Education (CDE), since they are an authoritative source of this data.

Why study LA County? LA County is the largest non-state government entity in the US and one of the most ethnically diverse US counties. It has a nominal GDP of over $700 billion, larger than Taiwan, Norway and Belgium, making it the third-largest metropolitan economy in the world. It has over 10 million residents and 88 municipalities, including its county seat of Los Angeles, which is the largest city in the state of California, and the second-most populous city in the US.

Data Quality Assessment

I considered the accuracy and completeness of the data. I could not assess the accuracy of the CDE chronic absenteeism statistics by census tract, since I could not examine the processes used to collect and tabulate them. The ACS survey data generally consists of population counts accompanied by counts of individuals that meet particular criteria, e.g. having a bachelor's degree. From these counts, I manually computed percentages using Google Sheets to use as predictor variables. These counts, and in a few cases, precomputed percentages, came with confidence intervals that could be used to assess the likely accuracy of the survey data. However, I chose not to do this, since I was interested in the percentages only as centers of probability distributions for use as indicator variables, not as predictors of discrete quantities.

To check for outliers that could indicate inaccurate data, I visualized the predictor and target variables using scatterplots with regression lines and a histogram in CARP-EXP Jupyter notebook. While there are certainly some outliers, I did not see any obviously erroneous data points.

To assess data completeness, the CARP-ETL Jupyter notebook counts the rows with missing or numerically invalid data from each data source, and creates a file that the CARP-EXP notebook uses to create a barplot that visualizes the completeness of each data source. I found that the CDE chronic absenteeism data covered 84% of the census tracts within LA County, while the ACS surveys covered between 98-100% of the tracts covered by the CDE data.

Feature Engineering

To create a single table (Pandas dataframe) of labeled observations for modeling, I used CARP-ETL to combine 12 datasets, including one on chronic absenteeism from CDE and a dataset on each of 11 ACS survey subjects. I manually prepared the ACS survey data using the American Fact Finder tool and Google Sheets before uploading it to Watson Studio. From the ACS and CDE data, CARP-ETL extracts required rows and columns, cleans them, and joins them on the six-digit US census tract number that uniquely identifies each dataset row.

Since I chose to assemble my own dataset from twelve different sources, I began the data exploration and feature engineering process with manual steps which made me familiar with the data and reduced the complexity of the automated data engineering required. Then, I coded straightforward data profiling, cleansing and joining techniques to produce a consolidated dataset--one table with 44 columns--for further exploration and modeling. Future enhancements to this product could further automate feature engineering.

Algorithms

For the deep learning algorithm for this regression problem, I chose a four-layer feed-forward neural network with a 25-node input layer followed by a 17-node hidden layer, an 11-node hidden layer, and a single-node output layer. Even though I had engineered a total of 43 predictor variables, I found that the neural network performed better when I restricted the number of predictor variables, and to the 25 most relevant, which CARP-ETL computes as the coefficient of determination (r^2) score of the predictor and target variables. I experimented with neural networks of various depths, but I found that any additional layers beyond the 4 I have described slowed down training without improving predictive accuracy. I sized the two hidden layers--the minimum required for deep learning--by using a rule of thumb which sets each layer's dimension as ⅔ the dimension of the previous layer plus the dimension of the output layer, which is one for regression problems.

Also, after extensive experimentation with a range of options, I selected weight initialization with a random normal distribution and RELU activation for each layer, and the Adam, optimizer with a mean squared error loss metric (see Model Performance Indicator section below). Even with optimal parameters, the training jobs converged to a useful result less than half the time, so I added early stopping logic to abort and restart when convergence was not occuring. See the CARP-DNN Jupyter notebook for additional details and hyperparameters.

For the non-deep learning algorithm I chose AdaBoost with a decision tree weak learner and a squared error loss metric (see Model Performance Indicator section below), after getting poor results with single-stage algorithms, such as linear regression and a standalone decision tree, as well as with other decision tree ensembles, such as Random Forest. In the CARP-DTE Jupyter notebook, the AdaBoost implementation tunes each decision tree in favor of those training examples that are evaluated inaccurately by previous trees, and combines each decision tree output into a weighted sum. Since decision trees inherently prioritize all predictor variables by the amount of entropy they reduce, I left in all 43 predictor variables instead of prioritizing them as I did when modeling with a deep neural network in CARP-DNN.

Frameworks

In choosing frameworks, I looked for the best and simplest tool for the work at hand. I also considered compatibility and ease of use within the IBM Watson Studio environment, the popularity of each framework, since I expect this work to be continued by others, and, finally, my own familiarity with the toolsets. I chose Python-based tools because of their overwhelming popularity and compatibility with IBM Watson studio. On top of Python, I layered a number of popular and well-supported choices:

- Numpy for scientific calculations and numerical manipulations
- Pandas for manipulation of labeled data tables, i.e. dataframes
- GeoPandas for manipulation of spatial dataframes, i.e. those that contain a specialized geometry column that can describe individual or multiple points, lines, and polygons
- Matplotlib for data visualization
- Seaborn layered on top of matplotlib for simplified data visualization
- Descartes for creating the choropleth map in CARP-EXP.

- Keras with a TensorFlow back-end for declarative neural network specification and straightforward execution of the resulting configurations, with a broad selection of network layer types, optimizers, activation functions, loss metrics and callbacks.
- Scikit-Learn for non-neural network modeling. While this framework is straightforward and very popular, it has the disadvantage of not supporting parallel processing. However, since this project dealt with a universe of less than 3,000 observation, Scikit-Learn was the simplest choice for the job. If this product is expanded in the future, it may be advisable to substitute a more scalable framework.

Model Performance Indicator

For the primary model performance indicator, I chose the Coefficient of Determination or r^2, which is a number between 0 and 1 that denotes the proportion of variance in a dependent variable that is explained by the variance in an independent variable. I used r^2 both for measuring relationships between individual independent variables and the target variable (chronic absenteeism percentage by census tract), as well as to determine overall model performance by comparing vectors of predicted and actual variables.

When used for pairwise variable measurements (rather than with multiple regressors, where it is known as R^2), r^2 yields a straightforward measurement of explained variation. Also, r^2 is conveniently proportional to the squared loss functions available with the Scikit-Learn implementation of AdaBoost and the mean squared loss function available with the Keras implementation of the Adam neural network optimizer.

Finally, in general, all performance indicators based on squared error have the advantages of aggregating all errors without errors of opposite sign canceling each other out, and of penalizing large errors disproportionately more than small ones. Both of these qualities are important in the prediction of chronic absenteeism rates, where a reliably close estimate can be used to shape mitigation policy and execution.

Additional Information

Enterprise Data

There is no need for enterprise data here, since this product is to be used by data scientists and data analysts to predict chronic absenteeism based on socioeconomic conditions, and to explore how chronic absenteeism can be mitigated.

Streaming Analytics

This product uses historical data to make predictions. There is no need for streaming analytics.

Data Integration

This product uses manual methods for data download, in order to allow users to change the data sources. It uses open-source Python-based tools for transforming and blending data, in order to allow as many people as possible to use the tool. Data transformation using these tools is straightforward and efficient.

Data Repository

This product uses standard the IBM Watson Data Platform object storage, since it can be easily manipulated by the Jupyter notebooks

Discovery and Exploration

This product uses standard Python-based tools for discovery and exploration, since I am familiar with them and I can use the same Python-based tools for implementation.   Also, their popularity supports additional contributors to the open-source project I will create.

Actionable Insights

This product uses standard Python-based tools to generate actionable insights, since they can generate the visualizations and tables necessary to present the insights my product will generate.

Data Products

The data product is a set of well-documented Jupyter notebooks with all data they use that can be deployed on any Jupyter server with the right open source libraries installed or available via an Internet connection. This will enable the widest range of users to benefit from this product.

Security, Information Governance and Systems Management

This product uses only public data. I plan to make this software free and open source, with a license that requires acknowledgement but allows derivative products. I will deploy it on Github.