UCLA Extension

Introduction to Data Science COM SCI X 450.1

Instructor: Daniel D. Gutierrez

CLASS PROJECT

Supervised Machine Learning - Regression

Overview

The class project is designed for you to become productive with your new data science skills by working though a real-life problem using a *California Housing* data set. We'll use the data set to predict *median house values*. This data set appeared in a 1997 paper titled Sparse Spatial Autoregressions by Pace, R. Kelley and Ronald Barry, published in the *Statistics and Probability Letters* journal. The researchers built it using the 1990 California census data. It contains one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). The variables found in the data set are as follows (the names are fairly self-explanatory, and all variables are atomic class numeric with the exception of `ocean_proximity` which is a factor):

`longitude`

`latitude`

`housing_median_age`

`total_rooms`

`total_bedrooms`

`population`

`households`

`median_income`

`median_house_value`

`ocean_proximity`

Each row pertains to a group of houses representing medians for groups of houses in close proximity.

The goal for this project is for you to gain experience in trying out the principles of data science using R as discussed in class. You'll need to carry out each of the tasks outlined below that parallel the *Data Science Process* detailed in class.

1. Access the Data Set

The first step is to access the data set and load it into the R environment. Follow these steps in order to complete this step:

a) Download the `housing.csv` data set available in Canvas. The data set has 20,640 rows and 10 variables.
b) Read the data set into R using a data frame named `housing`. Please do not use RStudio's data import feature, but rather write R code for accessing the data.
c) Cast the `ocean_proximity` character variable to the factor class, and display the resulting levels.

2. EDA and Data Visualization

Now, let's do some exploratory data analysis (EDA) and visualizations. Follow these steps below and provide commentary for what you find:

a) Run the `head()` and `tail()` functions on the data frame to get a feel for the actual data values.
b) Run the `summary()` function on the data frame to get a sense for the data classes, range of values for numeric variables, and any NAs found.

c) Perform a correlation analysis on numeric variables in the data frame.
d) Create histograms for each numeric variable.
e) Produce bloxplots for each numeric variable.
f) Produce boxplots for the variables: `housing_median_age`, `median_income`, and `median_house_value` "with respect" to the factor variable `ocean_proximity`.

3. Data Transformation

The next step is to transform the raw data into a more refined form as indicated in the steps below (that will constitute your *data pipeline*):

a) We see from the `summary()` results above that there are many `NA` values in the `total_bedrooms` variable (the only variable with missing values). This needs to be addressed by filling in missing values using *imputation*. You can use the "statistical median" for missing `total_bedrooms` values. The median is used instead of mean because it is less influenced by extreme outliers. This may not be the best method, as these missing values could represent actual buildings (e.g. a warehouse) with no bedrooms, but imputation often makes the best of a bad situation. You can use the `impute()` function covered in class, or write code to accomplish the requirement.
b) Split the `ocean_proximity` variable into a number of binary categorical variables consisting of 1s and 0s. Although many machine learning algorithms in R can handle categorical data stored in a factor variable, but we will cater to the lowest common denominator and do the splitting ourselves. Once you're done with the splitting, you can remove the `ocean_proximity` variable from the data frame.
c) Use the `total_bedrooms` and `total_rooms` variables along with `households` to create two new variables: `mean_bedrooms` and `mean_rooms` as these are likely to be more accurate depictions of the

houses in a given group. You can then remove the `total_bedrooms` and `total_rooms` variables once you've accomplished this requirement.

d) Perform *feature scaling*. Scale each numerical variable except for `median_house_value` (as this is our response variable), and the binary categorical variables.
e) The result of your data transformation efforts should yield a new data frame named `cleaned_housing` with the following variables:

```
"NEAR BAY"            "<1H OCEAN"           "INLAND"
"NEAR OCEAN"          "ISLAND"              "longitude"
"latitude"            "housing_median_age" "population"
"households"          "median_income"       "mean_bedrooms"
"mean_rooms"          "median_house_value"
```

4. Create Training and Test Sets

Now we can get ready for machine learning by creating the training and test sets using a random sample index.

a) Create a random sample index for the `cleaned_housing` data frame.
b) Create a training set named `train` consisting of 70% of the rows of the `cleaned_housing` data frame.
c) Create a test set named `test` consisting of 30% of the rows of the `cleaned_housing` data frame.

5. Supervised Machine Learning - Regression

In this step, you'll use the `randomForest()` algorithm found in the `randomForest` package for training and inference. Our goal is to predict the median house value using regression methods.

First, you'll need to separate your training set `train` into two pieces: `train_x` and `train_y` where `train_x` is a data frame that has all variables except the response variable `median_house_value` and `train_y` is a numeric vector

(not a data frame) that has only the response variable values from `median_house_value`. In class we used the "formula" method for calling machine learning algorithms, but some algorithms (another other languages) require you to pass the data separately: response variable and predictors, so this exercise will be good practice.

Next, you'll call the `randomForest()` algorithm, passing to it both components of the training set created above. Make sure you specify arguments `ntree=500`, and `importance=TRUE`. Return the resulting model in the object variable `rf` as in:

```
rf = randomForest(x=train_x, y=train_y ,
                          ntree=500, importance=TRUE)
```

Now use `names(rf)` to see all the different metrics computed by the algorithm.

6. Evaluating Model Performance

The final step of the project involves determining the quality of the fit for the statistical model, i.e. evaluating the performance of the random forest algorithm used above.

a) Calculate the *root mean squared error* (RMSE) for the trained model. You can use the object element `rf$mse` for this purpose. This element is a vector of calculated *mean squared error* (MSE) values, one for each "tree" used in the algorithm. You can use the last MSE value, i.e. the last element of the vector. Be sure to calculate the square root of this value to obtain the RMSE. The resulting RMSE is your prediction of median price of a house in a given district to within a RMSE delta of the actual median house price. This can serve as your benchmark moving forward as you experiment with other statistical models.
b) Next, we can see how well the model makes predictions by using the test set. Split the test set in the same manner as the training set above, creating

a new data frame `test_x` and numeric vector `test_y`. You can use the `predict()` function using the trained model `rf` along with `test_x` in order to calculate a vector of predicted median house values.

c) The last step is to calculate the RMSE for the test set using the vector of *predicted* median house values and the *actual* values from the test set, `test_y`. To calculate RMSE you can use the UDF we described in class, or you can write your own code.
d) How does the test set RMSE compare with the training set RMSE? Did the model score roughly the same on the training and testing data, suggesting that it is not overfit and that it makes good predictions?
e) Run the variable importance plot with the `varImpPlot()` function for the random forest model and discuss what it indicated about your feature vector (you may choose to re-train your model using only the feature variables suggested by this plot.

7. Project Report

Write a project report designed to communicate to a non-technical project stakeholder the results of the project. Please include the following sections:

- Project overview: goals and hypothesis. Communicate the "business question."
- Description of the data set and all variables.
- Discussion of results for your EDA and data visualization experiments (please include graphic images of all the plots in this section).
- Description of your data pipeline: data munging step.
- Description of the statistical model you used (random forest) including your feature vector and response variable.
- Description of the performance metric results for the model.
- Communicate the "business answer" for the project.

Please provide your report in PDF format.

8. Project Submission

Please use Canvas to submit two files for grading purposes: your R script, and the project report.

9. Congratulations!

Great job, you've just gone through the entire *Data Science Process* to create a supervised machine learning model to predict the response variable `median_housing_value`. You now have data science "super powers!"