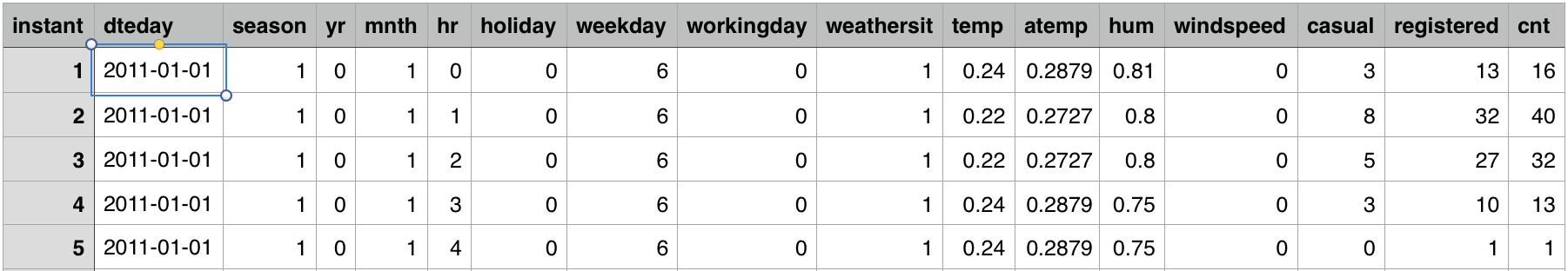
Many American cities have communal bike sharing stations where you can rent bicycles by the hour or day. Washington, D.C. is one of these cities. The District collects detailed data on the number of bicycles people rent by the hour and day.

[Hadi Fanaee-T](http://www.liaad.up.pt/area/fanaee) at the [University of Porto](http://www.up.pt/) compiled this data into a CSV file, which you'll be working with in this project. The file contains 17380 rows, with each row representing the number of bike rentals for a single hour of a single day. You can download the data from the [University of California, Irvine's website](http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset). If you need help at any point, you can consult the solution notebook [in our GitHub repository](https://github.com/dataquestio/solutions/blob/master/Mission213Solution.ipynb).

Here's what the first five rows look like:



Here are the descriptions for the relevant columns:

* instant - A unique sequential ID number for each row
* dteday - The date of the rentals
* season - The season in which the rentals occurred
* yr - The year the rentals occurred
* mnth - The month the rentals occurred
* hr - The hour the rentals occurred
* holiday - Whether or not the day was a holiday
* weekday - The day of the week (as a number, 0 to 7)
* workingday - Whether or not the day was a working day
* weathersit - The weather (as a categorical variable)
* temp - The temperature, on a 0-1 scale
* atemp - The adjusted temperature
* hum - The humidity, on a 0-1 scale
* windspeed - The wind speed, on a 0-1 scale
* casual - The number of casual riders (people who hadn't previously signed up with the bike sharing program)
* registered - The number of registered riders (people who had already signed up)
* cnt - The total number of bike rentals (casual + registered)

In this project, you'll try to predict the total number of bikes people rented in a given hour. You'll predict the cnt column using all of the other columns, except for casualand registered. To accomplish this, you'll create a few different machine learning models and evaluate their performance.

Instructions

* Use the [pandas](http://pandas.pydata.org/) library to read bike\_rental\_hour.csv into the dataframe bike\_rentals.
* Print out the first few rows of bike\_rentals and take a look at the data.
* Make a histogram of the cnt column of bike\_rentals, and take a look at the distribution of total rentals.
* Use the [corr](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html" \t "_blank) method on the bike\_rentals dataframe to explore how each column is correlated with cnt.

It can often be helpful to calculate features before applying machine learning models. Features can enhance the accuracy of models by introducing new information, or distilling existing information.

For example, the hr column in bike\_rentals contains the hours during which bikes are rented, from 1 to 24. A machine will treat each hour differently, without understanding that certain hours are related. We can introduce some order into the process by creating a new column with labels for morning, afternoon, evening, and night. This will bundle similar times together, enabling the model to make better decisions.

Instructions

* Write a function called assign\_label that takes in a numeric value for an hour, and returns:
  + 1 if the hour is from 6 to 12
  + 2 if the hour is from 12 to 18
  + 3 if the hour is from 18 to 24
  + 4 if the hour is from 0 to 6
* Use the [apply](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.apply.html) method on series objects to apply the function to each item in the hrcolumn.
* Assign the result to the time\_label column of bike\_rentals.

Before you can begin applying machine learning algorithms, you'll need to split the data into training and testing sets. This will enable you to train an algorithm using the training set, and evaluate its accuracy on the testing set. If you train an algorithm on the training data, then evaluate its performance on the same data, you can get an unrealistically low error value, due to overfitting.

Instructions

* Based on your explorations of the cnt column, pick an error metric you want to use to evaluate the performance of the machine learning algorithms. Explain why you chose this metric in a markdown cell.
* Select 80% of the rows in bike\_rentals to be part of the training set using the [sample](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sample.html" \t "_blank)method on bike\_rentals. Assign the result to train.
* Select the rows that are in bike\_rentals but not in train to be in the testing set. Assign the result to test.
  + This line will generate a Boolean series that's False when a row in bike\_rentalsisn't found in train: bike\_rentals.index.isin(train.index)
  + This line will select any rows in bike\_rentals that aren't found in train to be in the testing set: bike\_rentals.loc[~bike\_rentals.index.isin(train.index)]

Now that you've done some exploration and manipulation, you're ready to apply linear regression to the data. Linear regression will probably work fairly well on this data, given that many of the columns are highly correlated with cnt.

As you learned in earlier missions, linear regression works best when predictors are linearly correlated to the target and also independent -- in other words, they don't change meaning when we combine them with each other. The good thing about linear regression is that it's fairly resistant to overfitting because it's straightforward. It also can be prone to underfitting the data, however, and not building a powerful enough model. This means that linear regression usually isn't the most accurate option.

You'll need to ignore the casual and registered columns because cnt is derived from them. If you're trying to predict the number of people who rent bikes in a given hour (cnt), it doesn't make sense that you'd already know casual or registered, because those numbers are added together to get cnt.

Instructions

* Create a list of predictor columns to use in training and predictions.
  + At a minimum, this list should exclude the cnt, casual, dteday, and registeredcolumns.
  + Feel free to remove other columns you don't think will be useful for the predictions.
* Use the [LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \t "_blank) class from sklearn to train a machine learning algorithm on train.
  + Use only the columns in the list of predictors for this.
* Make predictions using the [LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \t "_blank) class on test.
  + Use the exact same predictor columns here.
* Calculate the error between the predictions and the actual values.
* Write your thoughts about the predictions and the error in a markdown cell.

Now you're ready to apply the decision tree algorithm. You'll be able to compare its error with the error from linear regression, which will enable you to pick the right algorithm for this data set.

Decision trees tend to predict outcomes much more reliably than linear regression models. Because a decision tree is a fairly complex model, it also tends to overfit, particularly when we don't tweak parameters like maximum depth and minimum number of samples per leaf. Decision trees are also prone to instability -- small changes in the input data can result in a very different output model.

Instructions

* Use the [DecisionTreeRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html" \t "_blank) class to fit a decision tree algorithm to the train data.
* Make predictions using the [DecisionTreeRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html" \t "_blank) class on test.
* Calculate the error between the predictions and the actual values.
* Experiment with various parameters of the [DecisionTreeRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html" \t "_blank) class, including min\_samples\_leaf, to see if it changes the error.
* Write your thoughts on the predictions and the error in a markdown cell.

You can now apply the random forest algorithm, which improves on the decision tree algorithm. Random forests tend to be much more accurate than simple models like linear regression. Due to the way random forests are constructed, they tend to overfit much less than decision trees. Random forests can still be prone to overfitting, though, so it's important to tune parameters like maximum depth and minimum samples per leaf.

Instructions

* Use the [RandomForestRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html" \t "_blank) class to fit a random forest algorithm to the traindata.
* Make predictions using the [RandomForestRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html" \t "_blank) class on test.
* Calculate the error between the predictions and the actual values.
* Experiment with various parameters of the [RandomForestRegressor](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html" \t "_blank) class, including min\_samples\_leaf, to see if it changes the error.
* Write your thoughts on the predictions and the error in a markdown cell.

That's it for the guided steps! We recommend exploring the data more on your own.

Here are some potential next steps:

* Calculate additional features, such as:
  + An index combining temperature, humidity, and wind speed
* Try predicting casual and registered instead of cnt.

We recommend creating a [GitHub](https://github.com/) repository and placing this project there. It will help other people, including employers, see your work. As you start to put multiple projects on GitHub, you'll have the beginnings of a strong portfolio.

You're welcome to continue working on the project here, but we recommend downloading it to your computer using the download icon above and working on it there.