Applications of Classification Trees to San Diego Marine Layer Prediction

By Laura Thapa
Advisors: Michael Holst and Joel Norris
June 6th, 2018

Abstract

This paper evaluates the usefulness of classification trees, a machine learning algorithm, as a tool for predicting the behavior of the marine layer in San Diego from late spring to early fall. First, we provide background information about the marine layer and machine learning and discuss some motivation for applying machine learning to weather prediction problems. We then implement a classification tree to answer the question: given certain meteorological conditions in the early morning, will the marine layer burn off by noon? Finally, we use cross tabulation analysis to compare the classification tree model to a persistence forecast model and a climatological mean forecast model.

Table of Contents

1. Introduction

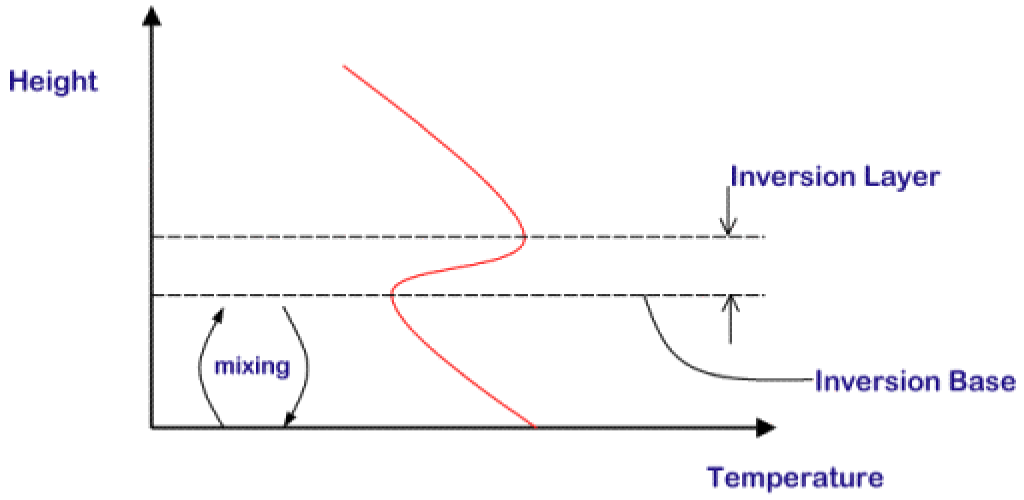
1.1 Marine Layer Overview

The following, including figures, are drawn from [1].

The marine layer is a phenomenon in which a temperature inversion forms over the ocean and low clouds are unable to escape from beneath this inversion. This creates the famous “June gloom” that every San Diegan is familiar with.

A temperature inversion is a section of the atmosphere in which the temperature increases with height. This is different form the normal atmospheric lapse rate, in which temperature decreases with height. This temperature inversion creates a “lid” on the lower air; it does not allow surface air to travel past the inversion base. See Figure 1.

Figure 1—Idealized temperature inversion profile.



Temperature inversions form as a result of high pressure and cold surface air. The marine layer temperature inversion in Southern California is a combination of the pacific high and coastal upwelling. The pacific high is a region of high pressure over Southern California that forces warm air towards the surface. Coastal upwelling is a process in which currents along the California coast cause cold deep water to be pulled up to the surface. This in turn cools the air near the surface. At a certain height in the atmosphere, the warm air being forced down meets the cold air near the surface. A temperature inversion forms at this interface because as height increases, the temperature is transitioning from cold to warm.

Marine layer clouds are a combination of two factors: the mixing of moist air below the temperature inversion, and the spatial relationship between the lifting condensation level and the inversion base height. Moist air trapped under the inversion layer is able to mix freely, and if this air reaches a height called the lifting condensation level (LCL), a cloud will form. Marine layer clouds form on days when the LCL is below the inversion base height (IBH), as in Figure 2. These clouds are then trapped between the LCL and the IBH. However, as in Figure 3, there are days where the LCL is above the IBH, and marine layer clouds are unable to form. Therefore, “June gloom” days are often characterized by high relative humidity, LCL below IBH, and a strong temperature inversion.

Figure 2—IBH > LCL. Marine layer clouds form and get trapped between the LCL and the IBH

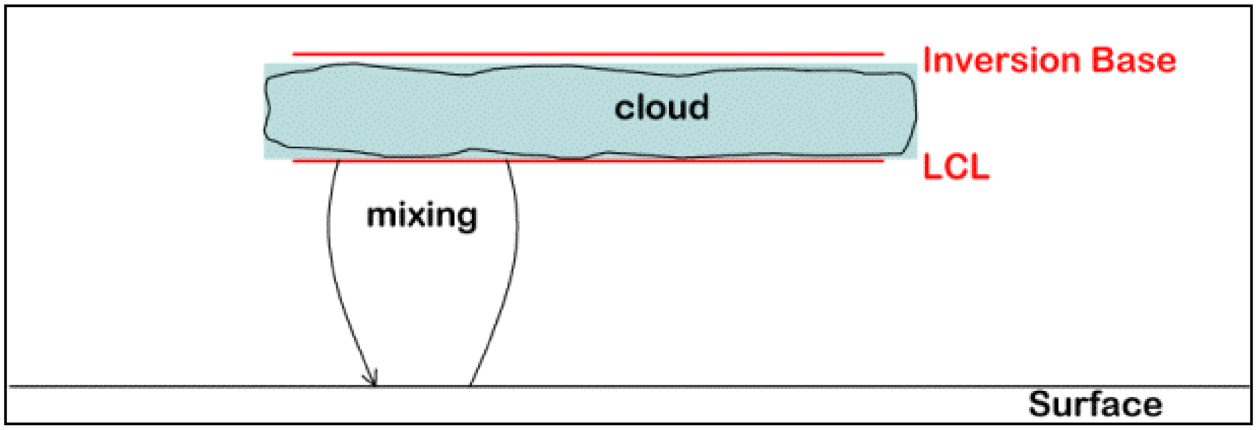
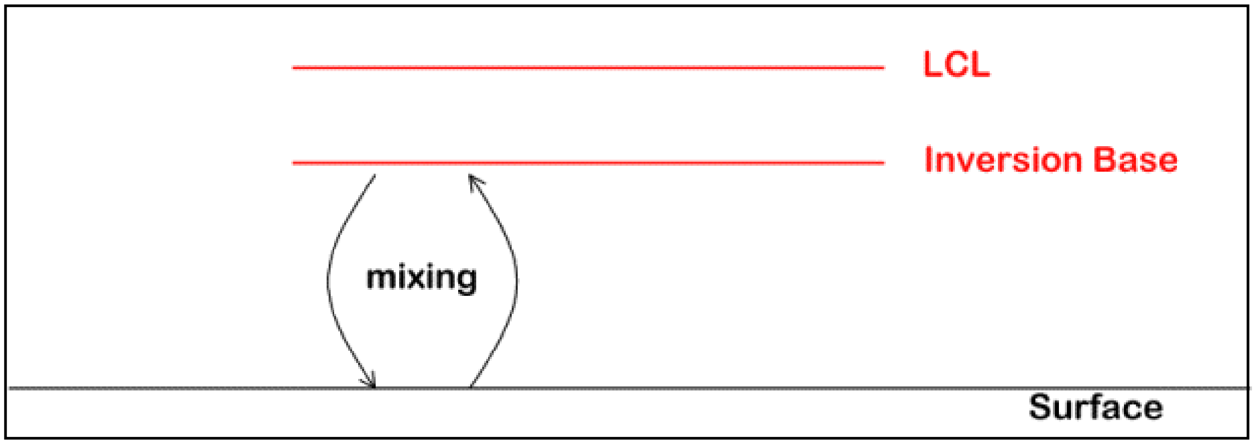


Figure 3—IBH<LCL. Marine layer clouds are unable to form.



The marine layer "burns off" when the sun rises. This causes two processes to happen, both of which lead to decreased relative humidity and thus to cloud burnoff. When the sun rises, the lower air warms. Since warm air can "hold on" to more water vapor, the relative humidity of the warmer air is lower, even though the same amount of water vapor may be present. Once the relative humidity drops, this causes the LCL to rise, making it harder for low clouds to form. Sunrise and the accompanying warming of the lower air also weakens the temperature inversion. This allows warm, dry air from above to mix into the low clouds. This also serves to decrease relative humidity and raise the LCL. These two processes work together until all of the marine layer clouds have dissipated.

The time it takes for marine layer clouds to burn off depends on the thickness of the clouds and the strength of the temperature inversion. Thicker clouds means more warm and dry air has to mix in in order for clouds to dissipate. A stronger temperature inversion means that it takes more to warm up the lower air enough to start the processes that lead to cloud dissipation.

1.2 Machine Learning Overview

This section draws heavily on [2].

What is machine learning?

Machine learning is a branch of computer and data science that deals with teaching computers to recognize patterns in given data and make predictions based on new data. There are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. This paper will focus on supervised learning, which is when the programmer explicitly tells the computer which outcomes it must predict.

Supervised Machine Learning

Supervised machine learning happens in several steps. The first step is training, in which the programmer provides the computer with input and output data. The machine learning algorithm then builds a model based on the given data. The goal in the training phase is to find the best way to map inputs to outputs.

The next step is called validation. This stage is about evaluating the performance of the model that was built in the training phase. This is done by handing the model inputs with known outputs and then allowing the model to create outputs of its own. The programmer must then determine how well the model performed. The validation phase is also a good time for the programmer to make any changes to the model in order to improve model performance.

The final step is testing. This is when the model makes predictions about real world data with unknown outputs.

Frequently in supervised learning, the inputs are called predictors or features, and the outputs are called response or target variables.

Classification and Regression

Supervised machine learning deals with classification and regression algorithms. Classification models group outputs into distinct bins, whereas regression models allow for output values to be continuous.

What are some common machine learning algorithms?

Some common supervised algorithms include: linear regression, random forest, gradient boosting, support vector machines, and decision trees. This paper will focus mainly on decision trees. For more detailed information, see [2, algorithm overview].

Linear Regression

Linear regression in machine learning works in much the same way as linear regression in statistics or numerical analysis. It seeks to find the function of the form:

$$y = ax + b$$

which will best fit a collection of points. It does this by least squares or gradient descent to minimize the error between the points and the line.

Support Vector Machine (SVM)

Support Vector Machine is an algorithm in which each data point is a point in $\boldsymbol{R}^n$, where n is the number of predictors. These points are then plotted in n-dimensional space called the feature space, and the SVM finds the optimal hyperplane that either splits the points into groups (classification) or is closest to all the points (regression). For example, as in [3], the SVM tries to find a function of the form:

$$f(x) = \big(w.\phi(x)\big) + b$$

by calculating coefficients w and b and choosing nonlinear function ϕ.

Decision Trees

Decision trees split data sets into smaller groups that are homogeneous internally and distinct from each other. Since decision trees will be discussed in much greater depth in the later sections of this paper, for now I will just mention that they use the gini index, entropy, and chi squared statistic as well as weighted averages of these to split the population according to the most important variable.

Random Forest (RF)

A random forest is a collection of decision trees, each of which gets a chance to classify new points and then "vote" for which class the points should belong to. The class with the most votes is the class that the point is assigned to. A basic algorithm for creating a random forest is as follows:

1. Train the trees, each on a random sample of size N that is taken from the total data set with replacement.
2. Set a number m<M, where M is the number of predictors, and randomly select m predictors to try to split with respect to at each node.
3. Trees grow as large as possible; pruning is not allowed.
4. Voting process to classify new data

Gradient Boosting (GB)

According to [2], gradient boosting is used when there are a lot of predictors and the programmer wants an algorithm with "high predictive power". As explained in [4], gradient boosting is an algorithm that tries to find a function $y = f(x)$ that will minimize a loss function. This function is approximated as a weighted sum of "weak learners", which are predictors or predictive algorithms, according to [2]. The weighted sum is notated in [4] as:

$$f(x) = \sum_{m=0}^{M} \beta_m h(x, \theta_m)$$

where β_m and θ_m are chosen based on the fitting the training set and $h(x, \theta_m)$ are decision trees. In a more general sense, $h(x, \theta_m)$ can be any algorithm with "low predictive power".

2. Machine Learning and the Marine Layer

2.1 Machine Learning vs. GCMs

Normally, climate scientists use General Circulation Models (GCMs), Statistical-Dynamical Models, or Numerical Weather Prediction to study trends in the Earth's climate. These models apply the Navier-Stokes equations, thermodynamics concepts, and statistical techniques to the Earth as a whole and evaluate them over time steps.

The marine layer is a local phenomenon rather than a global one, so GCMs are not the right tool for predicting its behavior. Examining some differences between GCMs and NWPs and machine learning algorithms can make it clear why machine learning may be a good fit for studying the marine layer. The first difference between more traditional weather prediction models and machine learning models is the underlying math. GCMs and NWPs rely on partial differential equations, whereas machine learning algorithms rely on statistics and linear algebra. The other important difference is the problem of resolution. GCMs are unable to resolve local weather phenomena accurately without the aid of a supercomputer. With machine learning algorithms, on the other hand, one can study precise locations using a laptop. In conclusion, it is worth trying out machine learning techniques for local weather because they simplify the underlying math and can give good results without using a supercomputer.

2.2 Significant Papers

Applying machine learning to weather and climate problems is not a new idea. The three papers that I examined looked at using machine learning algorithms to predict temperature, marine layer clouds, humidity, and pressure. They used a variety of techniques, including Support Vector Machines (SVMs), Random Forest (RF), Gradient Boosting (GB) and linear regression. These three papers justify applying machine learning algorithms to local weather problems, and provide some inspiration regarding specific methods I can use to tackle predicting marine layer behavior.

The first paper [4] deals with an application of SVM, RF, and GB to the problem of the marine layer and solar irradiance. This paper concludes that machine learning algorithms did a better job than physics based numerical weather prediction models. An additional conclusion is the fact that GB and RF are able to pick out the most important variables for prediction.

The second paper [3] evaluates the usefulness of Support Vector Regression in predicting daily maximum temperatures. The conclusion is that SVMs perform better than artificial neural networks (ANNs) and could replace ANNs in some applications.

The final paper [5] discusses how machine learning outperforms traditional numerical weather forecasting. The argument is that machine learning techniques are "robust to perturbations", meaning that small changes in initial data lead to only small changes of output data. The paper analyzes a linear and a functional regression technique and concludes that linear regression works better over a shorter (2 days) time span of inputs than functional regression.

These papers were useful because they illustrated that machine learning algorithms can have meaningful applications to weather problems. In particular, [4] showed me that decision trees (the components of a Random Forest) are useful in predicting the marine layer's behavior. There are many factors that influence marine layer cloud burn off, and I thought it would be interesting to see which factors the decision tree considered the most important. [5] was useful because weather, and thus the inputs to the predictive model, is always changing, so it makes sense to use an algorithm that can accurately predict outputs for a wide range of inputs.

3. Decision Trees

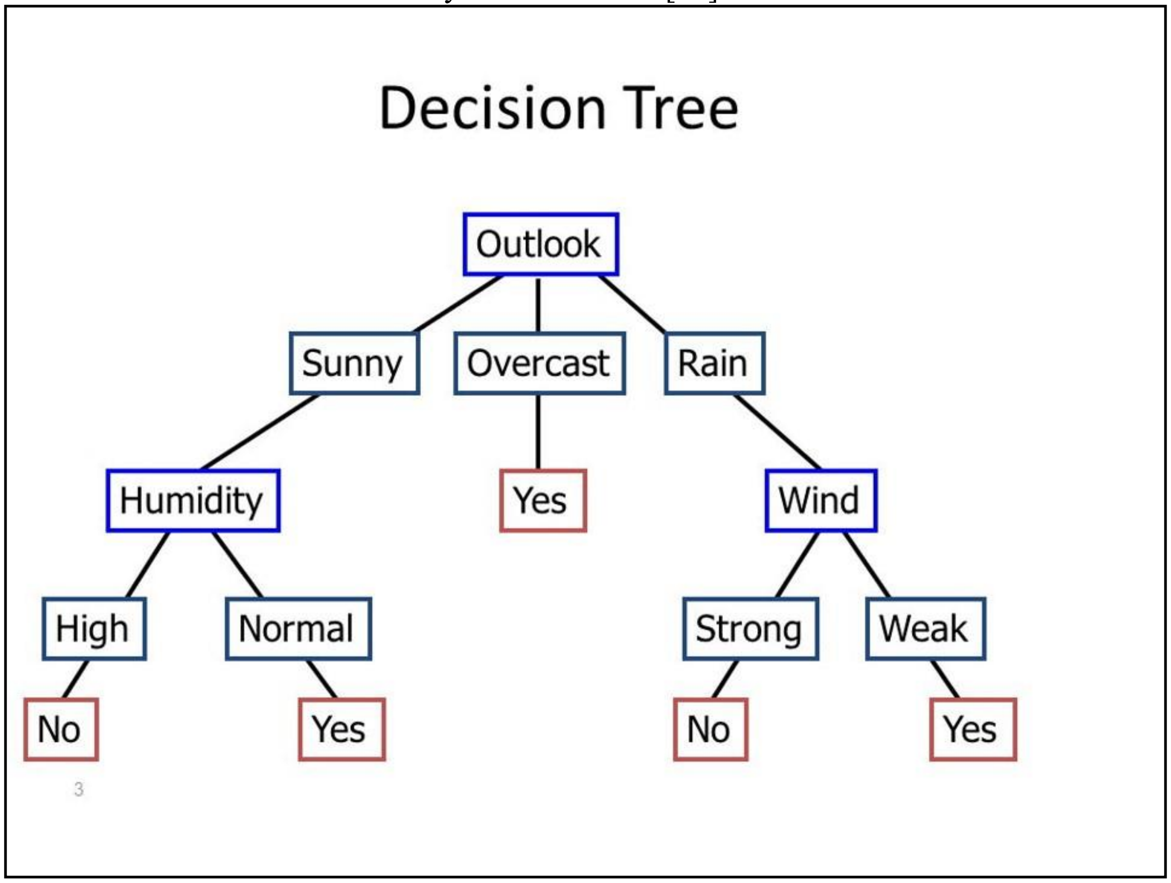
3.1 Decision Trees Overview

A decision tree is a kind of supervised classification algorithm in which the whole training data set is divided into smaller and smaller groups that are similar internally and distinct from each other. You can think of the outcome of a decision tree as a kind of flow chart that if you were to put a new data point into it, you could classify it based on one characteristic at a time (see figure 4).

Some important terms relating to decision trees are as follows:

Node—Group of data points in a decision tree
Predictor/Feature—Characteristic of a data point
Response—Characteristic the programmer hopes to predict
Split—Division of a node according to a certain predictor
Root node—Node at the top of the decision tree. It contains all the data points in the training set.
Parent node—Node that will be split into smaller nodes
Sub node—Nodes that result from a split
Leaf/terminal node—Node at the bottom of the decision tree. Leaf nodes should be homogeneous internally with respect to the response variable, but distinct from each other with respect to the predictor variables.
Depth—Longest path from the root to a leaf. Defined exactly this way in [5]

Figure 4—Example decision tree model found online. In this case, the response variable is a binary (yes/no) answer to the question: Should I go outside? And the predictors are: outlook, humidity and rain. From [10].



3.2 Pros and Cons of Decision Trees

Decision trees are one of many machine learning algorithms, so what makes them useful for studying the marine layer? One reason to use decision trees is that they are easy to visualize. This means that the programmer can actually produce a picture of what the tree looks like, complete with all the splits and how the splits were created. Another reason that decision trees are useful in the marine layer problem is that they can predict categorical and continuous outcomes. In the context of predicting marine layer dissipation, that means decision trees can predict specific burn off times as well as whether or not the clouds will be gone by a certain time.

Another reason to use decision trees in local weather problems like the marine layer is because decision trees can pick out which predictor is the most significant. There are many factors in determining marine layer burn off time, and I thought that might be interesting to know what the algorithm thought was the most important variable in determining dissipation.

One thing that the programmer must be aware of when using decision trees is overfitting. Overfitting is when a model predicts the training set well and does not generalize to predicting new data such as a test or validation set. However, data analysts have come up with methods to avoid overfitting that are at the programmer's disposal.

3.3 Building a Decision Tree

The following is heavily drawn from [2, Decision Tree Specifics].
Here is a bare bones algorithm for building a decision tree:

1. For each split, beginning at the root node and going all the way to the terminal nodes
    a. Split the parent node with respect to each predictor.
    b. For each predictor
        i. Use a splitting criterion to evaluate the split
    c. Split the parent node based on the most significant split as determined by the splitting criterion.
2. Repeat step 1 until a stopping criterion is reached

As can be seen above, building a decision tree is all about splitting up the training data set. There are three techniques for making classification tree splits: gini index, chi square, and entropy and information gain.

Gini Index

The first splitting criterion is the Gini Index. This measures the homogeneity of the new sub node, or the likelihood that if we picked two data points from the same sub node, they would have the same value for the response variable. The values p and q are defined as the probability of each target variable in the new node. Thus, Gini Index works only for binary response variables.

The following is an algorithm for how to split based on Gini Index:

1. For each predictor
    a. Split with respect to that predictor
    b. Calculate the Gini Index for each sub node with the formula: $g = p^2 + q^2$
    c. Calculate the weighted Gini Index with the formula: $weighted\ gini = a_1 g_1 + a_2 g_2$
2. Determine which predictor resulted in the highest weighted Gini Index, and split with respect to that predictor.

Note that using Gini Index as a splitting criterion will only produce binary splits.

Entropy and Information Gain

The second splitting criterion is entropy and information gain. This method measures how disorganized a node is, or how much information is needed to describe that node. Entropy is defined to be zero if the node is pure (all the data points are in the same class for the outcome variable) and 1 if the node is split 50/50.

The following is an algorithm for how to split based on entropy. Please note that p and q are defined the same way as for Gini Index:

1. For each predictor
    a. Split with respect to that predictor
    b. Calculate the entropy for each sub node with the formula:
    $e = -p\log_2 p - q\log_2 q$
    c. Calculate the weighted entropy with the formula:
    $weighted\ entropy = a_1e_1 + a_2e_2$
2. Determine which predictor resulted in the lowest entropy and split with respect to that predictor.
3. Information gain is then defined to be: $1 - entropy$

Note that entropy can make binary and nonbinary splits.

Chi Squared

The final splitting criterion is the chi squared statistic for each possible split based on predictor. It measures the statistical significance of differences between parent and sub nodes. Success and failure are defined the same way as p and q.

An algorithm for splitting based on chi squared is as follows:

1. For each predictor
    a. Split with respect to that predictor
    b. Calculate the chi squared for each node
        i. Calculate chi squared for success and failure
        ii. Chi squared formula: $X^2 = \sqrt{\frac{(O-E)^2}{E}}$
    c. Chi squared for the split = $\sum_{nodes} X^2$
2. Split with respect to the predictor with the highest X^2 value.

3.4 Overfitting

Data scientists use size restriction, pruning, and k fold cross validation in order to avoid over fitting in a decision tree. Size restriction can be thought of as setting maximums or minimums on certain aspects of the tree. For example, the programmer can set a maximum or minimum number of terminal nodes, splits or predictors considered in a split. The programmer can also set limits on the depth of the tree or the number of observations required to make a new node.

Pruning is when the tree "looks ahead" and gets rid of nodes that are not beneficial. This is accomplished by first making a tree with many levels, and then calculating net gain/loss of each path down the tree. The tree then gets rid of branches that have net loss. Pruning is generally considered "better" than just a decision tree alone.

K-fold cross validation is when the training set is divided into k subsets, and k-1 of them are used for training the model and 1is used to validate it. This process is repeated k times, and each subset takes a turn at being the validation set. The model's performance (percentage of correct predictions) is then averaged over the k trials.

4. Experiment

4.1 Overview

In the experiment, I trained a decision tree using MATLAB's classification learner on sounding and coastal low cloudiness data taken in May-September in the even years from 1996-2017. Then, once MATLAB had built the decision tree model, I used the model to make a prediction about marine layer burn off in May-September of the odd years from 1996-2017. Theoretically, this model should be able to predict whether or not the marine layer clouds over UCSD (latitude: 32.88 and longitude: =117.23) will dissipate by noon.

4.2 Inputs

In my background research into the marine layer, I determined that there were six important parameters that help determine marine layer burn off. They are as follows:

- Relative Humidity (RH): Amount of water is in the air parcel as a fraction of the amount of water in the air parcel when it is saturated. If RH = 1, there is a cloud.
- Temperature Inversion Strength: The change in temperature from the bottom to the top of the inversion.
- Lifting Condensation Level (LCL): The height that an air parcel must be lifted to to form a cloud.
- Inversion Base Height (IBH): The height where the temperature inversion begins.
- Morning Coastal Low Cloudiness (Morning CLC): A binary variable that shows whether or not there will be a cloud at a certain coastal coordinate at 5 am. 1 = yes cloud, 0 = no cloud.
- Cloud Thickness: Vertical height of marine layer clouds. Calculated as LCL-IBH. (Note that because of this definition, thickness < 0 means there is a cloud and more negative thickness means the cloud is thicker).
- Time: Year, month, day

These six parameters will be considered predictors in the decision tree algorithm. RH, LCL, IBH, and Morning CLC can be used to determine if there is a cloud present. Thickness and Inversion Strength are used to determine cloud persistence and therefore whether or not the cloud will burn off by noon.

The response variable will be:

- Noon Coastal Low Cloudiness (Noon CLC): A binary variable that shows whether or not there will be a cloud at a certain coastal coordinate at 12pm. 1 = yes cloud, 0 = no cloud.

4.3 Data

Processing the Data

All the data except the CLC data came from NOAA's Integrated Global Radiosonde Archive (IGRA). This experiment uses the IGRA derived data. This data was already processed and parameters relevant for my algorithm calculated. The sounding data is measured from the San Diego/Miramar MCAS station (IGRA code: USM00072293, METAR code: NKX). They send up radiosondes attached to weather balloons twice a day at 0 and 12 UTC. This project uses the 12UTC derived sounding data as predictors, because that is equivalent to 5am PDT. PDT is used here because I am studying the marine layer from May-September, when daylight savings time is in effect.

The IGRA data came as a very large text file, and contained soundings all the way back to the 1940s. Each sounding was in the form of a header line containing some of the calculated parameters I needed followed by the actual measurements of temperature, pressure, RH, etc., which the weather balloon had made on its way up.

In order to extract the LCL, all I had to do was write a MATLAB code that would pull out the header lines and from there pull out the LCL values and store them in a vector. I then stored this vector in a spreadsheet.

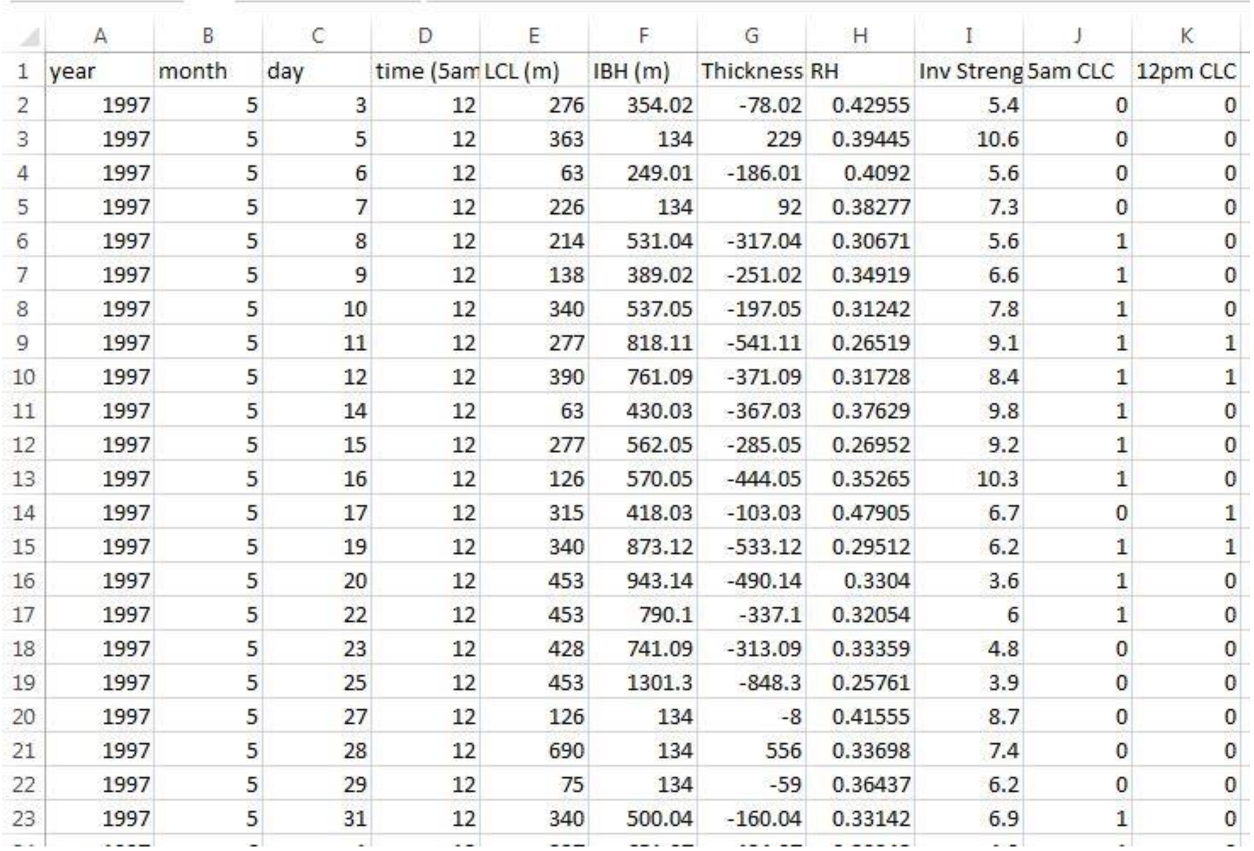
I had to work a little harder to extract the RH, IBH, and Inversion Strength. I obtained code from [6] which would take as inputs a vertical temperature profile vector, a height vector, and an elevation constant. I then wrote code that would extract the temperature and height vectors from the IGRA soundings (and convert them into the correct units). Then, I plugged these vectors into the code from [6] and obtained values for RH, IBH and Inversion Strength for each 12UTC sounding for May-Sept, 1996-2017. I then stored these vectors in the same spreadsheet.

The CLC data comes from [8], and it came in the form of a spreadsheet for each month for May-September of 1996-2017. Each spreadsheet can be thought of as a time x space matrix, with each row being a half hourly measurement in PST (so 1488 or 1440 rows depending on the month) and each column being a spatial coordinate (189446 columns corresponding to that many spatial coordinates). The data from [8] also included a file called COORDcoast that related specific coordinates to column numbers in the CLC matrix. For UCSD, which is at latitude: 32.88 and longitude: =117.23, this corresponded to a column number of 82796. Each element in the matrix was either 1 for yes cloud, 0 for no cloud, or -1 for insufficient data.

Since [8] gave the data in PST, I needed to extract the 4am and 11am PST columns to get the 5am and 12pm PDT data that I wanted. To do this, I wrote a combination of R and MATLAB code that would convert the data files from .rda to .csv and then extract the 4am and 11am cloudiness from the 82796th column. These vectors all went into the same spreadsheet where I had been storing everything else.

Finally, I used the spreadsheet to calculate the thickness as LCL-IBH. The final spreadsheet can be seen in Figure 5.

Figure 5—Final spreadsheet. Contains values for predictors and response variables in for May-September of 1996-2017.



	A	B	C	D	E	F	G	H	I	J	K
1	year	month	day	time (5am	LCL (m)	IBH (m)	Thickness	RH	Inv Streng	5am CLC	12pm CLC
2	1997	5	3	12	276	354.02	-78.02	0.42955	5.4	0	0
3	1997	5	5	12	363	134	229	0.39445	10.6	0	0
4	1997	5	6	12	63	249.01	-186.01	0.4092	5.6	0	0
5	1997	5	7	12	226	134	92	0.38277	7.3	0	0
6	1997	5	8	12	214	531.04	-317.04	0.30671	5.6	1	0
7	1997	5	9	12	138	389.02	-251.02	0.34919	6.6	1	0
8	1997	5	10	12	340	537.05	-197.05	0.31242	7.8	1	0
9	1997	5	11	12	277	818.11	-541.11	0.26519	9.1	1	1
10	1997	5	12	12	390	761.09	-371.09	0.31728	8.4	1	1
11	1997	5	14	12	63	430.03	-367.03	0.37629	9.8	1	0
12	1997	5	15	12	277	562.05	-285.05	0.26952	9.2	1	0
13	1997	5	16	12	126	570.05	-444.05	0.35265	10.3	1	0
14	1997	5	17	12	315	418.03	-103.03	0.47905	6.7	0	1
15	1997	5	19	12	340	873.12	-533.12	0.29512	6.2	1	1
16	1997	5	20	12	453	943.14	-490.14	0.3304	3.6	1	0
17	1997	5	22	12	453	790.1	-337.1	0.32054	6	1	0
18	1997	5	23	12	428	741.09	-313.09	0.33359	4.8	0	0
19	1997	5	25	12	453	1301.3	-848.3	0.25761	3.9	0	0
20	1997	5	27	12	126	134	-8	0.41555	8.7	0	0
21	1997	5	28	12	690	134	556	0.33698	7.4	0	0
22	1997	5	29	12	75	134	-59	0.36437	6.2	0	0
23	1997	5	31	12	340	500.04	-160.04	0.33142	6.9	1	0

Cleaning and Preparing the Data

My next step was to clean the data that I had extracted. First, I made sure that I had the same number of CLC measurements as I did IGRA measurements. Once that was done, I simply deleted days that had insufficient data. The two places where insufficient data occurred were in CLC and IBH (and hence thickness). I got rid of days that had -1 for their LCL value and also days that had N/A as their IBH value. I did this because I didn't think these days would give me meaningful predictions.

The next step was to split the data into a training set and a validation set. I decided to train the model on the data from the even years and then test the model on the data from the odd years. I did this so I could take advantage of any changes in the marine layer behavior over the past 20 years and maybe more accurately predict the odd years.

4.4 Running the Model

This decision tree was built with MATLAB's Classification Learner App, which is part of the Statistics and Machine Learning Toolbox. This app lets the user upload a data set and then use it to train a wide variety of models. It then assesses the performance of each model and assigns each model an accuracy percentage. This accuracy score is based on some internal validation (usually k-fold cross validation) that the app performs on the training set. If the user is not satisfied with the accuracy score, he or she can consider restricting/setting the number of splits or selecting which features the algorithm considers when making splits. After the user has optimized the model, he or she can then pick one or more models to output. These models are then exported to the MATLAB wokspace, and from there they can be used on testing or validation sets.

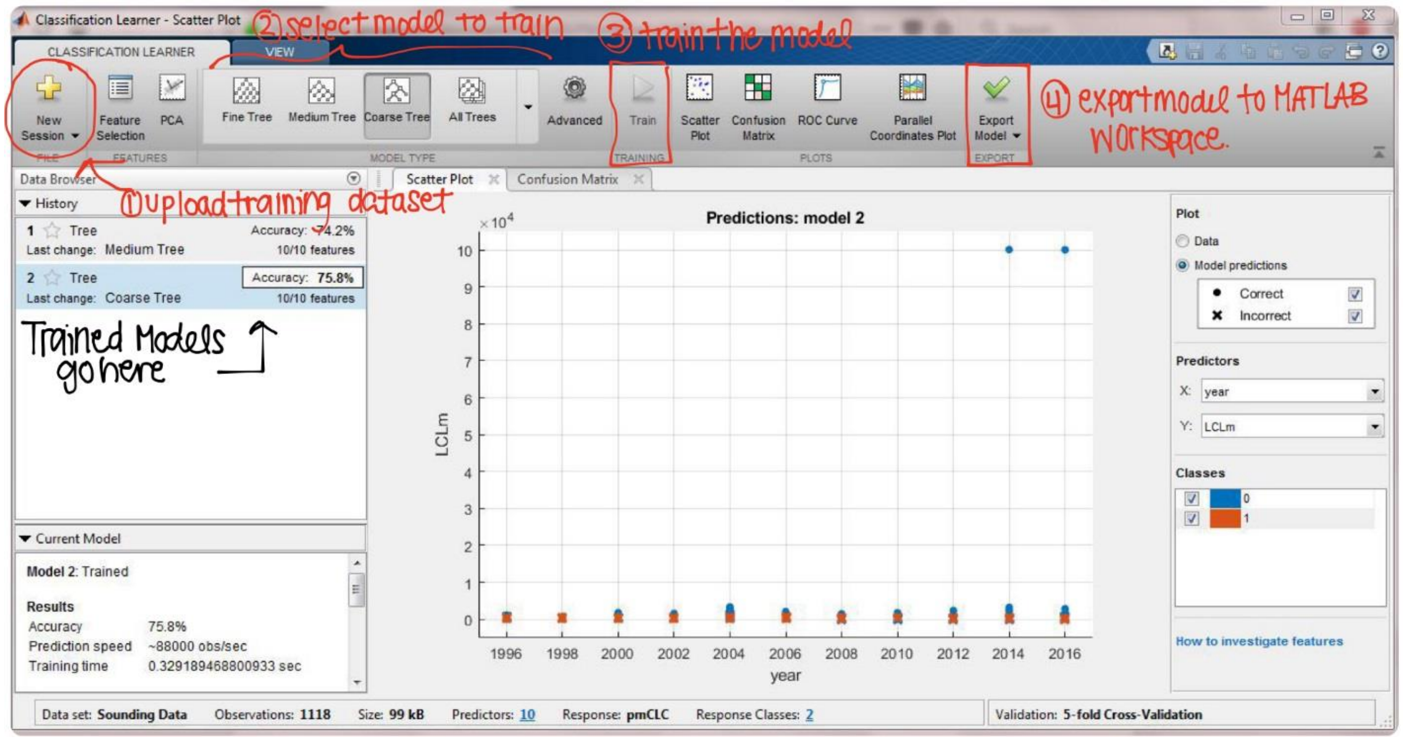
In this experiment, I trained a coarse tree and a medium tree on the even year data set, and then chose the coarse tree because it had a higher accuracy score. I did not perform pruning, size restriction, or feature selection in an attempt to improve the accuracy score. In this case, the validation scheme was 5-fold cross validation, which was chosen to prevent overfitting. See Figure 6.

After choosing the tree, I exported it into the MATLAB workspace. Then I uploaded the odd year data set, called it SoundingDataS4, and ran the following MATLAB code:

```
>> oddYearPredictions = trainedModel.predictFcn(SoundingDataS4);
>> view(trainedModel.ClassificationTree, 'Mode', 'graph')
```

The first line of this code makes applies the model that we created in the classification learner app to the validation data set (odd years) and makes a prediction for noon CLC. The output is a vector of ones and zeros. The second line of this code outputs a picture of the decision tree. See figure 7.

Figure 6—MATLAB's Classification Learner with the steps I performed to train and export the model



History

1	Tree	Accuracy: 74.2%
	Last change: Medium Tree	10/10 features
2	Tree	Accuracy: 75.8%
	Last change: Coarse Tree	10/10 features

Current Model

Model 2: Trained

Results

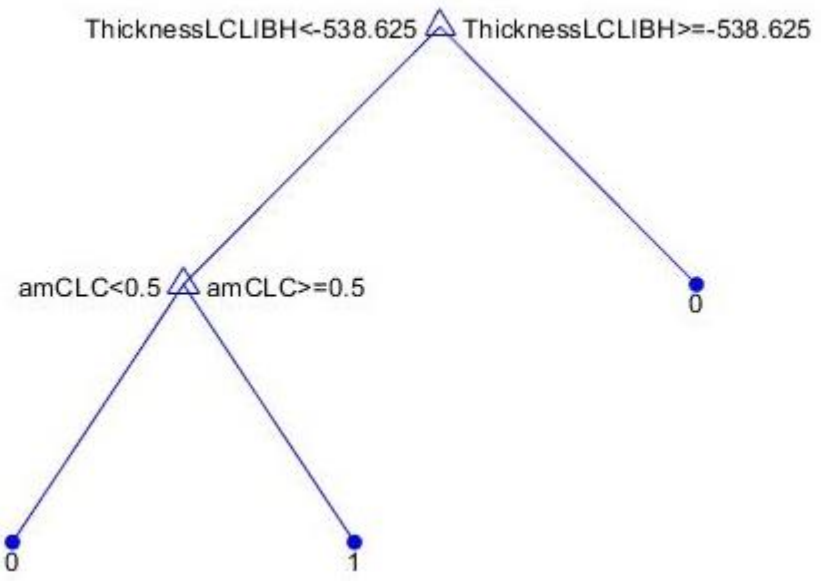
Accuracy	75.8%
Prediction speed	~88000 obs/sec
Training time	0.329189468800933 sec

Predictors X: year Y: LCLm

Classes: 0, 1

Data set: Sounding Data Observations: 1118 Size: 99 kB Predictors: 10 Response: pmCLC Response Classes: 2 Validation: 5-fold Cross-Validation

Figure 7—Decision tree built with the even years as the training set. The splits that were made show that the most important variables were cloud thickness and morning CLC. The idea is that clouds thinner than 538.25m will burn off by 12pm and that clouds thicker than 538.25m will persist if they were present in the morning but will not persist if there were no clouds in the morning.



5. Results

5.1 Persistence and Climatology

In order to analyze the performance of a forecast model, it must outperform two basic weather prediction techniques. These are persistence and climatology. If a model cannot do better than these two, it is not considered a very good model.

The persistence model says that whatever weather conditions are present today will also be present tomorrow. In the case of marine layer clouds, persistence says that if an observer sees low clouds one day, he or she will see low clouds the next day.

Climatology is based on long term averages for specific weather phenomena. It says that the forecast for any given day is the average of what that phenomenon has done over the past years. For the marine layer, it assumes that the marine layer clouds on any given day will burn off at the average of the burn off times over the past years. Another way to put this is that the predicted noontime cloudiness will behave like the average of past noontime cloudiness.

5.2 Cross Tabulation Analysis

As mentioned in [9], cross tabulation analysis is a technique that tallies up occurrences of a nominal variable, that is to say a variable grouped by name rather than number. A cross tabulation, also called a contingency table, has outcomes of one variable on one axis and outcomes of another variable on the other. The entries in the table are the times each combination of outcomes (one from each variable) occurs. Contingency tables are normally used to determine if there is an association between two variables, and they are often used in hypothesis testing for independence.

In this paper, I use contingency tables to compare the performance of a decision tree based forecast with the performance of a persistence forecast and a climatology forecast. In each table, true class is given in the column label and predicted class is given in the row label. For example, the number in the cell in the first row and first column is the number of days that the predicted class and the observed class were both zero. Thus, the numbers in the diagonal cells represent days that were correctly predicted. Days with a predicted value of 1 and an actual value of zero will be called false alarm days. I will evaluate the three forecast methods (persistence, climatology, and classification tree) against the known noon CLC for the odd years by comparing the correct prediction rate and the false alarm rate. These are as follows:

$$\%Correct\ Prediction = \frac{\sum diagonal\ cell\ values}{total\ number\ of\ observations} x\ 100$$

$$\%False\ Alarm = \frac{value\ in\ the\ second\ row, first\ column}{total\ number\ of\ observations} x\ 100$$

And ideally, the correct correction rate should be 100% and the false alarm rate should be 0%.

5.3 Analysis Process

The first step in the analysis process was to create a persistence forecast based on the assumption that noon CLC conditions from the previous day would persist into the next day. To do this, I wrote MATLAB code that took the noon CLC data as a vector input and outputted a vector that shifted each entry in the input down by one. The first entry was left unchanged; I made the assumption that the first day's conditions were persistent from the day before, for which I had no available data. The persistence forecast was made for the odd years.

Next, I wrote MATLAB code to simulate a climatology forecast. This was different form the persistence forecast in that it predicted only the 2017 summer. I did this because climatology forecasts rely on long term averages. First, I calculated the average cloudiness for each day, using the years form 1996-2016. Then, I used a random number generator to generate a value in the interval (0, 1). If this randomly generated value was below the average value for the day, then I predicted a 1 for noon time cloudiness. If the randomly generated value was above the average value for the day, then I predicted a 0 for no noontime cloudiness.

The next step was to perform the cross tabulation. For the decision tree and persistence forecasts, this was relatively easy to do. I started out with three vectors, which are as follows:

oddYearObservation—the vector of actual observations for the odd years from [8]
oddYearPersistence—vector obtained from the persistence forecast code
oddYearPrediction—vector obtained from the decision tree model

To perform the cross tabulation, I simply plugged these vectors into MATLAB's crosstab function, as follows:

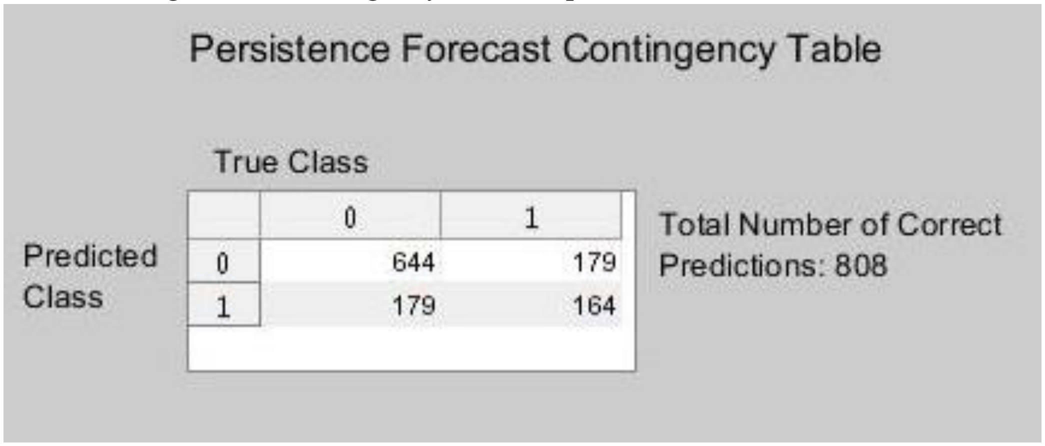
```
>> [tbl,chi2,p,labels] = crosstab(oddYearObservations, oddYearPersistince)
>> [tbl,chi2,p,labels] = crosstab(oddYearObservations, oddYearPrediction)
```

The first line of code cross tabulates the persistence model with the actual observations for the odd years, and the second one cross tabulates the classification tree based model with the actual observations for the odd years.

I had to work a little harder to get the contingency table for the climatology forecast. Since I was dealing with random numbers in creating the climatology forecast, what I ended up doing was creating multiple climatology forecasts and the associated contingency tables, and then taking the average of those tables.

After some manipulation with MATLAB's uitable command, which allows the user to turn a table into a figure and edit it, I obtained the contingency tables shown in Figures 8, 9, and 10.

Figure 8—Contingency table for persistence based forecast.



Persistence Forecast Contingency Table

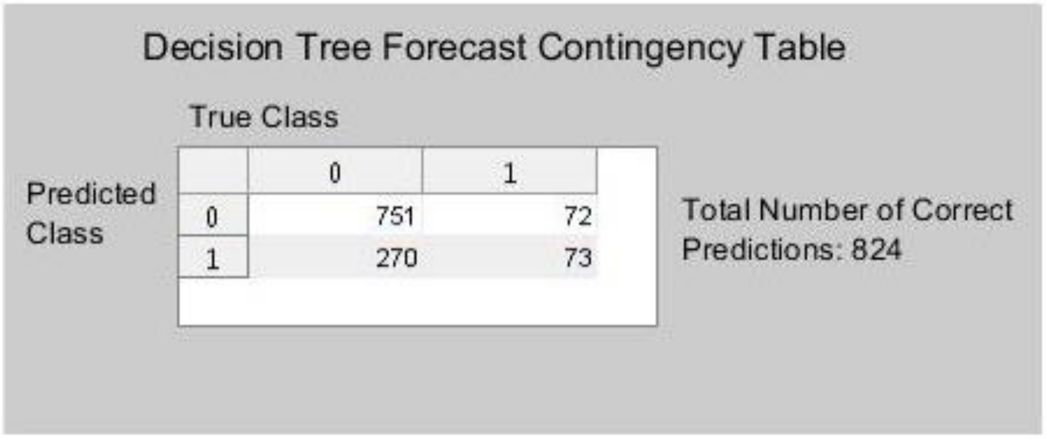
True Class

Predicted Class

	0	1
0	644	179
1	179	164

Total Number of Correct Predictions: 808

Figure 9—Contingency table for decision tree based forecast.



Decision Tree Forecast Contingency Table

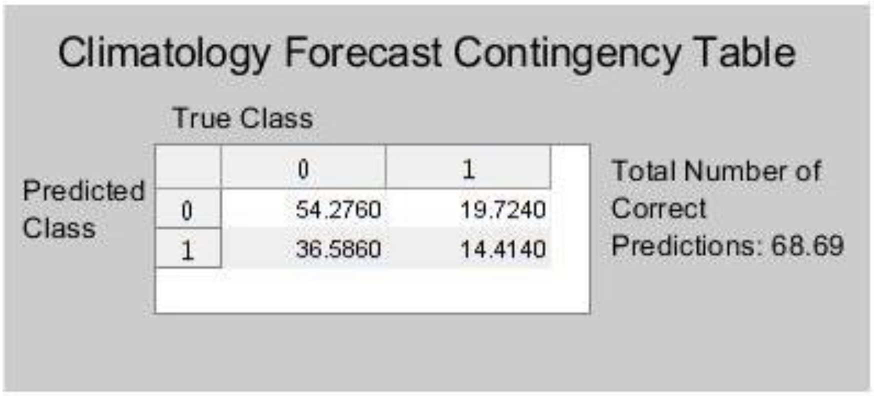
True Class

Predicted Class

	0	1
0	751	72
1	270	73

Total Number of Correct Predictions: 824

Figure 10—Contingency table for climatology forecast.



Climatology Forecast Contingency Table

True Class

Predicted Class	0	1
0	54.2760	19.7240
1	36.5860	14.4140

Total Number of Correct Predictions: 68.69

As mentioned above, I evaluate the performance of these models by checking the percentage of correct predictions and the false alarm rate for each forecasting method. The results are as follows:

$$\%correct_{persistence} = \frac{808}{1166} \times 100 = 69.3\%$$

and

$$\%correct_{climatology} = \frac{68.69}{125} \times 100 = 55.0\%$$

and finally

$$\%correct_{classification\ tree} = \frac{824}{1166} \times 100 = 70.7\%$$

Additionally,

$$\%false\ alarm_{persistence} = \frac{179}{1166} \times 100 = 15.4\%$$

and

$$\%false\ alarm_{climatology} = \frac{36.586}{125} \times 100 = 29.3\%$$

and finally

$$\%false\ alarm_{classification\ tree} = \frac{270}{1166} \times 100 = 23.3\%$$

From this one trial, this unmodified classification tree forecast model appears to outperform a persistence and climatology model in terms of correct predictions. It also has a better false alarm rate than climatology, but the false alarm rate for the classification tree is not as good as it is for persistence. I conclude that decision trees are worth pursuing further as a method of predicting marine layer behavior in San Diego.

5.4 Future Steps

Some future goals that I think are worth pursuing further would be to experiment with things like pruning and feature selection and see if these increase the accuracy of the decision tree as a forecasting method. Another interesting future problem would be to use a regression tree instead of a classification tree and see if a decision tree can correctly predict the time that the marine layer will burn off.

6. References

[1] Marine Layer Information and Figures 1, 2 and 3
http://meteora.ucsd.edu/~iacob/marinelayer.html

[2] Analytics Vidhya
Algorithm Overview: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/
Decision Tree Specifics: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/
Extra information on Gradient Boosting: https://www.analyticsvidhya.com/blog/2015/05/boosting-algorithms-simplified/

[3] Y.Radhika and M.Shashi. *Atmospheric Temperature Prediction using Support Vector Machines*. International Journal of Computer Theory and Engineering, Vol. 1, No. 1, April 2009 1793-8201.

[4] Zhong, Xiaohui. *Advancing solar irradiance/marine layer stratocumulus forecasting in California,* 2017. UC San Diego Electronic Theses and Dissertations

[5] Mark Holmstrom, Dylan Liu, Christopher Vo. *Machine Learning Applied to Weather Forecasting,* 2016. Stanford University.

[6] Definition of depth of a decision tree: https://cs.stackexchange.com/questions/39679/size-of-decision-tree-and-depth-of-decision-tree

[7] Monica Zamora Zapata, PhD Student with Solar Resource Assessment & Forecasting Laboratory at UCSD

[8] Rachel E.S. Clemesha, PhD. Postdoctoral Scholar. Paper: Rachel E. S. Clemesha, Alexander Gershunov, Sam F. Iacobellis, A. Park Williams, and Daniel R. Cayan. *The northward march of summer low cloudiness along the California coast,* 2016. AGU Geophysical Research Letters.

[9] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing,* Second Edition, 1988-1992.

[10] The example decision tree in figure 4 was obtained here:
http://slideplayer.com/slide/7027851/