STAT 30270

Data Analysis Project Submission

Predict Solar Power System Coverage in US States

Student Number: 16422014

Contents

1 Abstract

DeepSolar is a solar installation database for the US, built by extracting information from satellite images. Photo voltaic panel installations are identified from over one billion image tiles covering all urban areas as well as locations in the US by means of an advanced machine learning framework. I have used the resulting data set to train an AdaBoost.M1 model to be capable of predicting if certain areas contain a high or low number of solar power systems based on social, economical and geographical factors. This model has the capacity to predict under-utilised areas and help companies deploy solar power systems in optimal locations. I hope this model can assist in locating areas where solar power systems are most justified or needed.

2 Introduction

Fossil fuel emissions affect social and environmental health – clean air, safe drinking water, sufficient food and secure shelter. The direct damage costs to health is estimated to be between USD 2-4 billion/year by 2030 [1]. After many decades of false starts, renewable energy is poised to one day challenge fossil fuels as the energy source of choice in America. For example, in 2018, 17.1% of electricity in the United States was generated from renewable sources [2]. Solar panels are one of the leading sources of renewable energy. Despite the high upfront cost they have little maintenance expenses, are extremely environmentally friendly and can be considered insurance against rising electricity prices. Currently there are too few companies or governments invested in using the valuable insights derived from the available data sets. By understanding the dynamics at play and predicting under-utilised optimal locations for solar panels we can influence homeowners to invest in renewable energy and mitigate these risks.

3 Supplementary Data and Features

I included a few additional feature variables to complement the data set provided. One of these is the latitude [3] of each state as I believe this would greatly impact the viability of certain locations. I also included a variable representing the number of state policies and incentives put in place relating to renewable energy [4]. I finally included the number of U.S. solar panel manufacturers that are located in each state given in the data set [5].

4 Data Visualisation

4.1 Solar Panel Coverage by State

I initially examined the relationship between the solar power systems and U.S. states.

```
data = read.csv('data_project_deepsolar.csv', header = T)

library(ggthemes)
library(ggplot2)

# Plotting Solar System Count by State
p <- ggplot(data[,1:2], aes(x=state, fill=solar_system_count)) +
  geom_bar() + my_post_theme + labs(fill = "Type") + xlab("Count") +
  theme(axis.title.x = element_blank()) +
  theme(plot.title = element_text(hjust = 0.5))
```

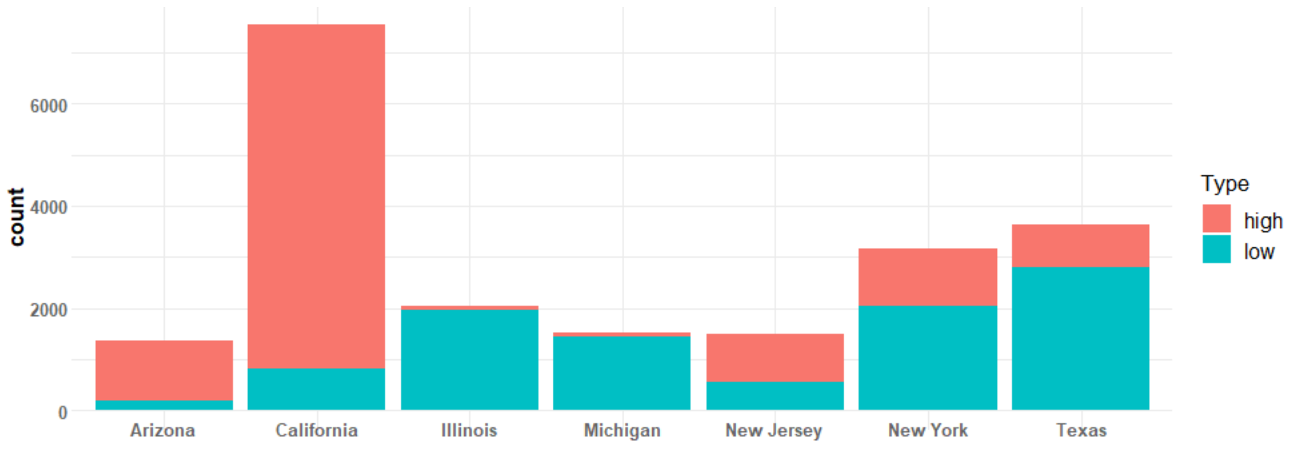

Figure 1: Solar Panel Counts by State

We can see that in the case of Illinois, Texas and Michigan a large proportion of their tiles have less than 10 solar power systems. Arizona and New Jersey may have less overall locations than the other states but their tiles tend to have more than 10 solar systems. It is also worth noting that Californian tiles make up a significant amount of the data set and also contain the highest proportion of tiles with more than 10 solar systems.

4.2 Wealth and Occupation Analysis

I wanted to drill down into these variables and see what correlations existed between wealth and solar power system density. I plotted continuous density plots for solar system counts against average household income, household unit value and occupation sector.

```
# Wealth and Occupation
cdplot(data$solar_system_count ~ data$average_household_income,
       xlim=c(min(data$average_household_income),250000), ylab='',
       xlab="Average Household Income", col = c("#F8766D", "#619CFF"))
```

```

cdplot(data$solar_system_count ~ data$housing_unit_median_value,
       xlim=c(min(data$housing_unit_median_value),1800000), ylab='',
       xlab="Median Housing Unit Value", col = c("#F8766D", "#619CFF"))

cdplot(data$solar_system_count ~ data$occupation_information_rate,
       xlim=c(min(data$occupation_information_rate),.2), ylab='',
       xlab="Proportion Employed in Information Sector", col = c("#F8766D", "#619CFF"))

cdplot(data$solar_system_count ~ data$occupation_construction_rate,
       xlim=c(min(data$occupation_construction_rate),.28), ylab='',
       xlab="Proportion Employed in Construction Sector", col = c("#F8766D", "#619CFF"))
```

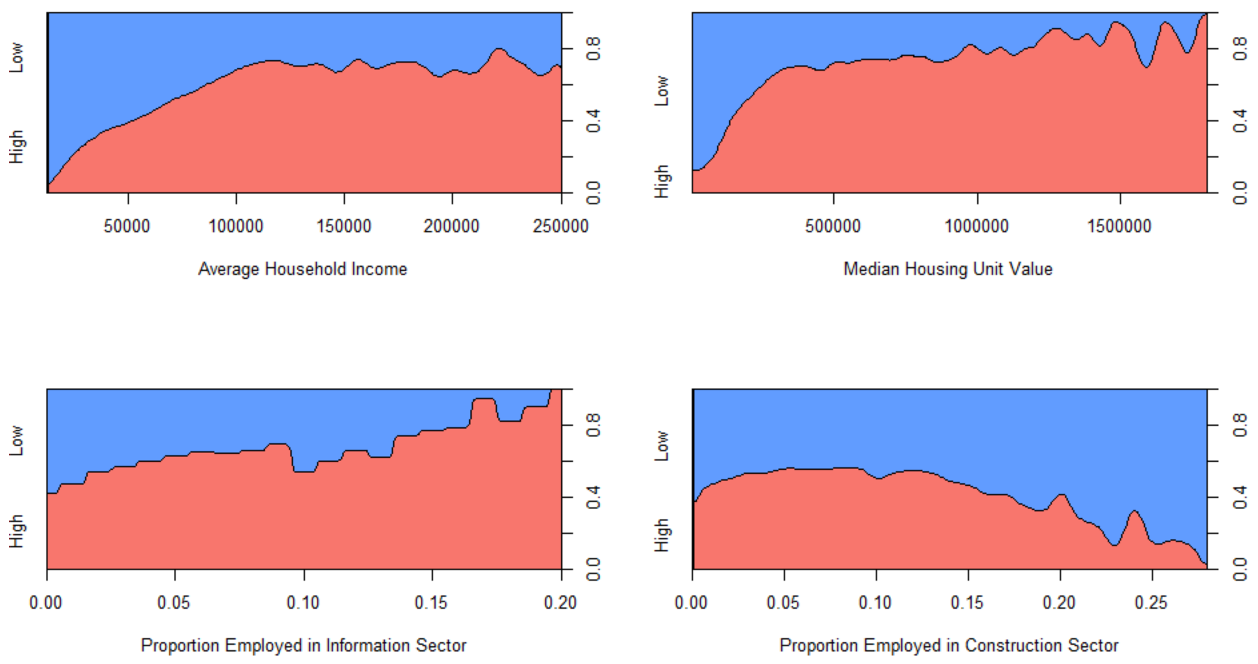

Figure 2: Solar Panel Counts by Wealth and Occupation

We can see that households with a higher average income tend to have more tiles with a high amount solar systems. We can also see that areas with a high proportion of information sector employees tend to have a larger amount of solar systems. These high income occupations, working in companies such as Facebook and Google, tend to be located in California which explains the increased solar panels in that state. Lower income workers, such as those in the construction sector tend to have far less solar tiles with a high level of solar systems.

It's also worth noting that the installation of solar panels is generally quite expensive and this may create adverse selection as solar panels may not allocated in the optimal location. Solar system count also increases with housing unit value. This is most likely due to the high cost of living in areas such as California. Another possibility is that these expensive houses are also quite large and thus, very suitable for solar panels tiles.

4.3 Political Effects

I was also curious to explore any possible political effects on solar system coverage. I plotted continuous density plots for solar system coverage against the proportion that voted for the Democratic party in the 2012 and 2016 general elections.

```
# Politics
cdplot(data$solar_system_count ~ data$voting_2016_dem_percentage,
       ylab='', xlab="Proportion voted Democratic 2016", col = c("#F8766D", "#619CFF"))
cdplot(data$solar_system_count ~ data$voting_2012_dem_percentage,
       ylab='',xlab="Proportion voted Democratic 2012", col = c("#F8766D", "#619CFF"))
```

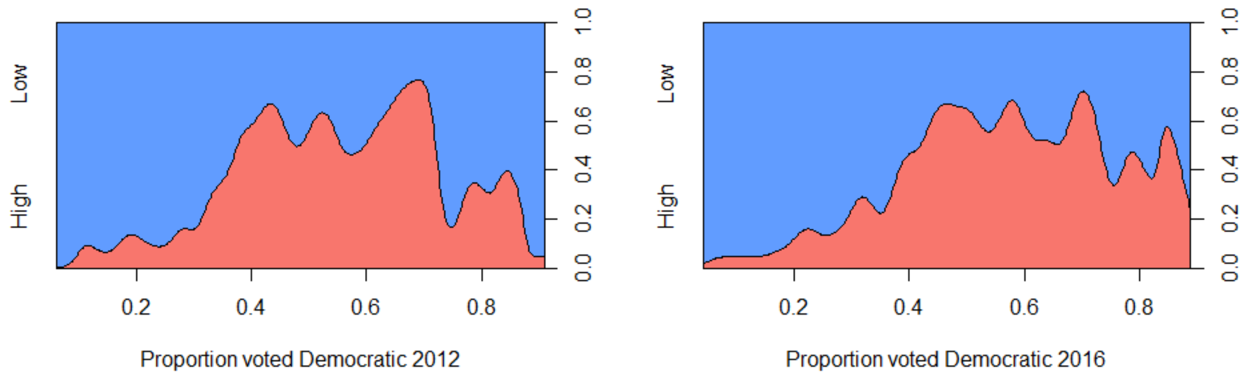

Figure 3: Political Effects on Solar Panel Counts

It is clear that as areas increase in democratic party support they also tend to increase in solar panel coverage. The steeper incline for the 2016 general election indicates this trend is more significant for the 2016 voting behaviours. This may be due to the attitude of the 2016 republican candidate in relation to global warming and climate change [6].

5 Methods

The required task is a binary classification between "high" or "low" amount of solar power systems in a given tile. The models I will consider for this task are Logistic Regression, AdaBoost, Bootstrap Aggregation tree, Random Forest algorithm and Support Vector Machines.

5.1 Data Preparation

5.1.1 Data Preprocessing

After confirming there were no missing values I proceeded to convert the variables related to democratic victories in 2012 and 2016 by label encoding values of 1 or 0.

```r
# Missing values
head(data)
sum(is.na(data)) # this is zero
Y <- data[,1]
X <- data[,3:84]
# Convert political win by label encoding
X$voting_2016_dem_win <-  as.numeric(X$voting_2016_dem_win)
X$voting_2012_dem_win <-  as.numeric(X$voting_2012_dem_win)
```

I then used one hot encoding to convert the state variable. This is because it has more than levels than the political victory factor and is not ordinal in nature.

```r
# One hot encode the state variable
library(mltools)
library(data.table)

onehot_state <- one_hot(as.data.table(data[,2]))
onehot_state
```

5.1.2 Feature Extraction

I currently have 84 predictor variables. To reduce the number of dimensions, I scaled the resulting data frame and performed principal component analysis. This is an operation that can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. Principal component analysis reduced the dimensionality of my training data significantly, leaving me with 49 linearly uncorrelated predictor variables.

```r
# Combine all variables
X <- cbind.data.frame(onehot_state, X)
# Fitting a principal component analysis
library(caret)
PCA <- preProcess(X, method = c("scale", "pca"), thresh = .99)
X_pca <- data.frame(predict(PCA, X))
```

5.1.3 Data Splitting

I split the cleaned data into 80% for training and 20% for any testing done after the model selection is complete. This 20% holdout set can be used to test model accuracy on out of sample data.

```r
# Split into training and holdout set
library(caTools)
clean_data <- cbind.data.frame(Y, X_pca)

split_data_ind <- sample.split(clean_data$PC2, SplitRatio = 0.8)
train_df <- subset(clean_data, split_data_ind == TRUE)
holdout_df <- subset(clean_data, split_data_ind == FALSE)
```

5.2 Model Training and Selection

5.2.1 Training Process

For initial model selection I implemented 10-fold cross validation during training of each model named above. K-fold cross validation divides the training data into K randomly selected, equal folds and then proceeds to train on (K-1) of these folds, testing on the remaining fold. The process is repeated for all permutations of the 10 folds.

5.2.2 Initial Model Selection

I first examined the mean training accuracy for each model during the 10-fold cross validation. Accuracy is a suitable metric in this context as the high:low target split is approximately 53:47.

```r
# Get table of mean accuracy
library(xtable)
out # the accuracy at each fold for each classifier
avg <- data.frame(t(round(colMeans(out), 3)))
print.xtable(xtable(avg), file = "./class_acc.txt")
```

Metric	AdaBoost	Log Regression	Bag Tree	Random Forest	SVM(Linear)	SVM(RBF)
Accuracy	0.901	0.881	0.853	0.853	0.878	0.897

After examining the above table it is clear that the logistic regression, bagging tree, support vector machine with linear kernel and random forest algorithms are under-performing relative to the other two models. From now on I will exclude them from the model selection process.

5.3 Model Tuning

I will now optimise each model with regards to their respective hyperparameters. I will use the same 10-fold cross validation to find optimal hyperparameters on the training data.

5.3.1 Support Vector Machine (RBF Kernel)

Apart from the kernel, the support vector machine has two main hyperparameters, the "cost" and "gamma". If the value of "cost" is large the model will use more data points to construct the support vector leading to higher variance and lower bias, which may lead to the problem of overfitting. The gamma parameter defines how far the influence of single training example reaches. If the value of gamma is high, then our decision boundary will depend on points close to the decision boundary and nearer points carry more weights than far away points.

```
# Optimise the SVM using training data
N = nrow(train_df)
K <- 10
folds <- rep(1:K, ceiling(N/K))
folds <- sample(folds) # random permute
folds <- folds[1:N] # ensure we got N data points

Cost = 2^(-1:4) # default is 1
Gamma = 2^(-8:-3) # default is 1/(data dimension)
options(max.print=1000000)
acc <- array(dim=c(length(Cost), length(Gamma), K),
             dimnames = list(Cost,Gamma, c(1:10)))

for (k in 1:K) {

  train_ind <- which(folds != k)
  test_ind <- setdiff(1:N, train_ind)

  for (c in 1:length(Cost)){

    for (g in 1:length(Gamma)){

      svm = svm(formula = Y ~ ., data = train_df, subset = train_ind, cost = Cost[c],
            gamma = Gamma[g], type = 'C-classification', kernel = 'radial')

      pred <- predict(svm, type = "class", newdata = train_df[test_ind,])
      tab <- table(train_df$Y[test_ind], pred)

      acc[c,g,k] = sum(diag(tab))/sum(tab) # store accuracy
    }
  }
  cat("Just finished fold number: ",k)
}

fold_avg_svm = apply(acc, c(1,2), mean)
which(fold_avg_svm == max(fold_avg_svm), arr.ind = TRUE) # Optimal is: Cost = 2, Gamma = 0.015625
```

After examining the results I concluded that the optimal pairing of hyperparameters (out of the ones I tested) were a Cost of 2 and Gamma of 0.015625.

5.3.2 AdaBoost

The AdaBoost.M1 algorithm contains three different algorithms in the adabag package; Freund's, Breiman's and Zhu's. These three equations alter the way in which the model updates the weight of observations after each iteration. I also have the ability to choose the number of iterations that are most efficient to avoid entering a period of overtraining. I split my training data set into 70% training and 30% validation. After approximately 200 iterations both algorithms seemed to converge so I have simplified the plot by removing the remaining 800 iterations.

```
# Optimise AdaBoost using training data
boost_split <- sample.split(train_df$PC2, SplitRatio = 0.7)
boost_train <- subset(train_df, boost_split == TRUE)
boost_test <- subset(train_df, boost_split == FALSE)

fitboost_Fre <- boosting(Y ~ ., data = boost_train, coeflearn = "Freund", mfinal=1000, boos = FALSE)
```

```r
Error_Fre <- errorevol(fitboost_Fre, boost_test)$error

fitboost_Bre <- boosting(Y ~ ., data = boost_train, coeflearn = "Breiman", mfinal=1000, boos = FALSE)
Error_Bre <- errorevol(fitboost_Bre, boost_test)$error

fitboost_Zhu <- boosting(Y ~ ., data = boost_train, coeflearn = "Zhu", mfinal=1000, boos = FALSE)
Error_Zhu <- errorevol(fitboost_Zhu, boost_test)$error

# Plot error paths, only first 200 iterations
mat <- cbind(Error_Fre[1:200], Error_Bre[1:200], Error_Zhu[1:200])
cols <- c("deepskyblue4", "darkorange3", "darkred")

matplot(mat, type = "l", col = cols,
        lwd = 2, xlab = "Number of trees",
        ylab = "Classification Error")
legend(x = 140, y = 0.17, cex = 0.75, legend = c("Freund", "Breiman", "Zhu"),
       col = cols, lwd = 2, bty = "n", title="Algorithm")
legend(x = 175, y = 0.17, cex = 0.75, col = cols, lwd = 2, bty = "n", title="Minimum",
       legend = c(round(min(Error_Fre), 4), round(min(Error_Bre), 4), round(min(Error_Zhu), 4)))
points(apply(mat, 2, which.min), apply(mat, 2, min),
       col = cols, pch = c(15,16,17), cex = 1.2)
```

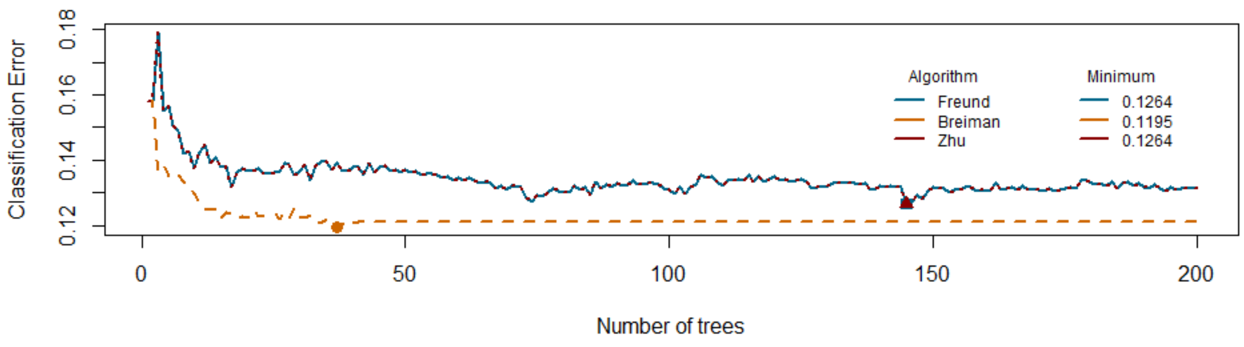

Figure 4: AdaBoost Error across Iterations

We can see that Breiman performs best, achieving a minimum generalised classification error of 0.1195 with 37 iterations. I will use this as my optimal AdaBoost implementation. However, we can see that Breiman's algorithm gets stuck after a certain number of iterations. This makes it theoretically possible for Freund or Zhu to reach a lower value after many more iterations (but they seem to have already converged so this is very unlikely).

5.3.3 Comparison

I compared the two optimal models over 100 replications of a 5-fold cross validation.

```r
## Compare both optimised models ##
# store winning model #
N = nrow(train_df)
K <- 5
R <- 100

acc_vals <- vector("list", R)

winner <- matrix(NA, R, K)

```

```r
for ( r in 1:R ) {
  acc <- matrix(NA, K, 2)
  folds <- rep( 1:K, ceiling(N/K) )
  folds <- sample(folds)
  folds <- folds[1:N]

  for ( k in 1:K ) {
    train_ind <- which(folds != k)
    test_ind <- setdiff(1:N, train_ind)

    # fitting
    svm_rbf = svm(formula = Y ~ ., data = train_df, subset = train_ind,
                  type = 'C-classification', kernel = 'radial', cost = 2, gamma = 0.015625)

    fitboost <- boosting(Y ~ ., data = train_df, subset = train_ind,
                         coeflearn = "Breiman", mfinal = 37, boos = FALSE)

    # predict
    pred <- predict(svm_rbf, type = "class", newdata = train_df[test_ind,])
    tab <- table(train_df$Y[test_ind], pred)
    acc[k,1] <- sum(diag(tab))/sum(tab)

    pred1 <- predict(fitboost, type = "class", newdata = train_df[test_ind,])
    tab1 <- table(train_df$Y[test_ind], pred1$class)
    acc[k,2] <- sum(diag(tab1))/sum(tab1)

    # select the best classifier
    winner[r,k] <- ifelse( acc[k,1] > acc[k,2], "SVM(RBF)", "Adaboost")
  }
  acc_vals[[r]] <- acc
  print(r)
 }

acc_vals
avg <- t(sapply(acc_vals, colMeans))

# find the mean accuracy across all iterations
meanAcc <- colMeans(avg)
meanAcc

# find stand.dev of these means
# applies sd function to the columns of avg
sdAcc <- apply(avg, 2, sd)/sqrt(R)
sdAcc

# beautiful plot of results
matplot(avg, type = "l", lty = c(2,3),
        col = c("darkorange2", "deepskyblue3"),
        xlab = "Replications", ylab = "Accuracy")

# add confidence intervals
bounds1 <- rep( c(meanAcc[1] - 2*sdAcc[1], meanAcc[1] + 2*sdAcc[1]), each = R )
bounds2 <- rep( c(meanAcc[2] - 2*sdAcc[2], meanAcc[2] + 2*sdAcc[2]), each = R )
polygon(c(1:R, R:1), bounds1,
        col = adjustcolor("darkorange2", 0.2), border = FALSE)
polygon(c(1:R, R:1), bounds2,
        col = adjustcolor("deepskyblue3", 0.2), border = FALSE)

# add estimated mean line
abline(h = meanAcc, col = c("darkorange2", "deepskyblue3"))

# add legend
legend("topright", fill = c("darkorange2", "deepskyblue3"),
       legend = c("SVM", "AdaBoost"), bty = "n")
```

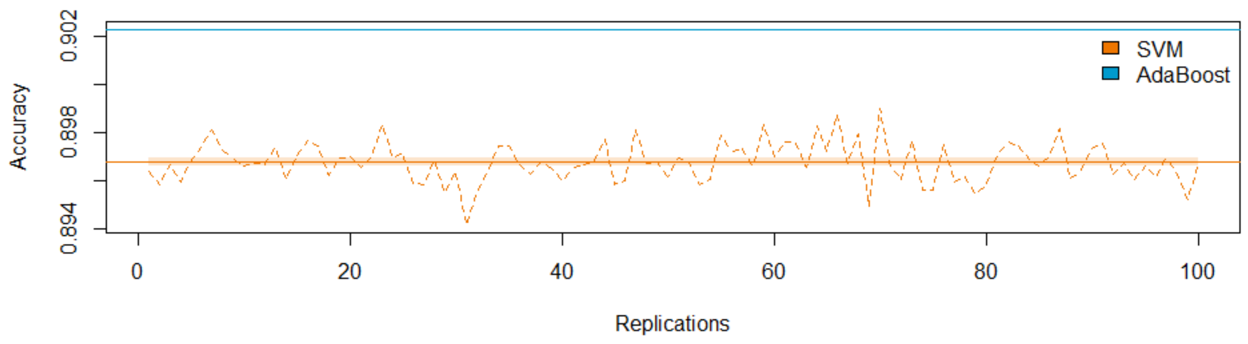

Figure 5: Accuracy across Replications

We can see clearly that the AdaBoost algorithm is superior due to consistently outperforming the SVM. During the 500 tests AdaBoost had the best accuracy score 94% of the time.

6 Results

I trained the AdaBoost model on the entire training data set and tested it on the holdout set.

Metric	Training	Testing
Accuracy	0.907	0.902

We can see the model has an extremely strong ability to predict data it has never seen before with an accuracy score over 90% on both the training and holdout data sets.

7 Discussion

I have shown that it is possible to create a machine learning model capable of classifying certain images as having a high or low count of solar systems based solely on the social, economic, geographical and meteorological data. This has implications for deployment and monitoring of renewable energy across the U.S. In general, these variables are much easier to obtain than satellite imagery making it possible to analyse the locality of solar power systems more efficiently.

A limitation of my method is the fact that each observation is examined independently. I do have access to the U.S. state but no other, more granular, location variables. It may be possible to incorporate some sort of dependent relationship which would capture the fact that high solar system counts will tend to be close to each other (or not). Also, I am sure that with more time and compute I could improve these accuracy results. I could more intensively grid search a larger space of hyperparameters and possibly consider other models such as artificial neural networks or extreme gradient boosting.

8 Conclusion

To conclude, I hope this can further aid policymakers and companies to deploy solar power systems in optimal locations throughout the United States. I also hope my analysis sheds some light on inequality and other sociocultural trends in solar deployment.

References

[1] World Health Organisation statements about climate change implications. `https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health`.

[2] Summary statistics about renewable energy in the United States. `https://en.wikipedia.org/wiki/Renewable_energy_in_the_United_States#Solar_power`.

[3] The latitude measurements for each U.S. state. `https://www.latlong.net/category/states-236-14.html`.

[4] The number of policies and rebates for renewable energy in each U.S. state. `https://www.dsireusa.org/`.

[5] The locations of U.S. solar panel manufacturers that produce solar panels for the traditional residential, commercial and utility-scale markets. `https://www.solarpowerworldonline.com/u-s-solar-panel-manufacturers/`.

[6] A variety of quotes by Donald Trump in relation to climate change and global warming. `https://edition.cnn.com/2017/08/08/politics/trump-global-warming/index.html`.