Predicting Forest Fires in Montesinho National Park - Portugal

Gabriel Estivalet

DS-SF-24

Aug/2016

Getting the data

Coordinates in a 10x10 grid

Numerical ratings for the level of moisture in different ground types

Relative humidity (%)

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0

Celsius

km/h

ml

len(data) = 517

Expected rate of fire spread

Burned area of a forest fire in ha

Data Dictionary

area: Burned area of a forest fire (ha) 0-1190 ha

X: x-axis coordinate of the Montesinho park map: 1 to 9

Y: u-axis coordinate of the Montesinho park map: 2 to 9

FFMC: A numerical rating of the moisture content of litter and other cured fine fuels: 18.7 to 96.2

DMC: A numerical rating of the average moisture content of loosely compacted organic layers and medium-size woody material: 1.1 to 291.3

DC: A numerical rating of the average moisture content of deep, compact, organic layers: 7.9 to 860.6

ISI: A numerical rating of the expected rate of fire spread: 0.0 to 56.10

Month: month of the year: 1 to 12

Day: day of the week: 1 to 7

Temp: temperature in Celsius degrees: 2.2 to 33.30

RH: relative humidity in %: 15.0 to 100

Wind: wind speed in km/h: 0.40 to 9.40

Rain: outside rain in mm/m2: 0.0 to 6.4

How can we predict if there will be a fire?

If there is a fire, what's the size of the area that is most likely be affected?

What are the most important variables that can help with our predictions?

Part I: Initial Analysis

Initial Analysis

Created dummy variables for month and day

```
data = Dummify(data, ['month'], del_prams = True)
data = Dummify(data, ['day'], del_prams = True)
```

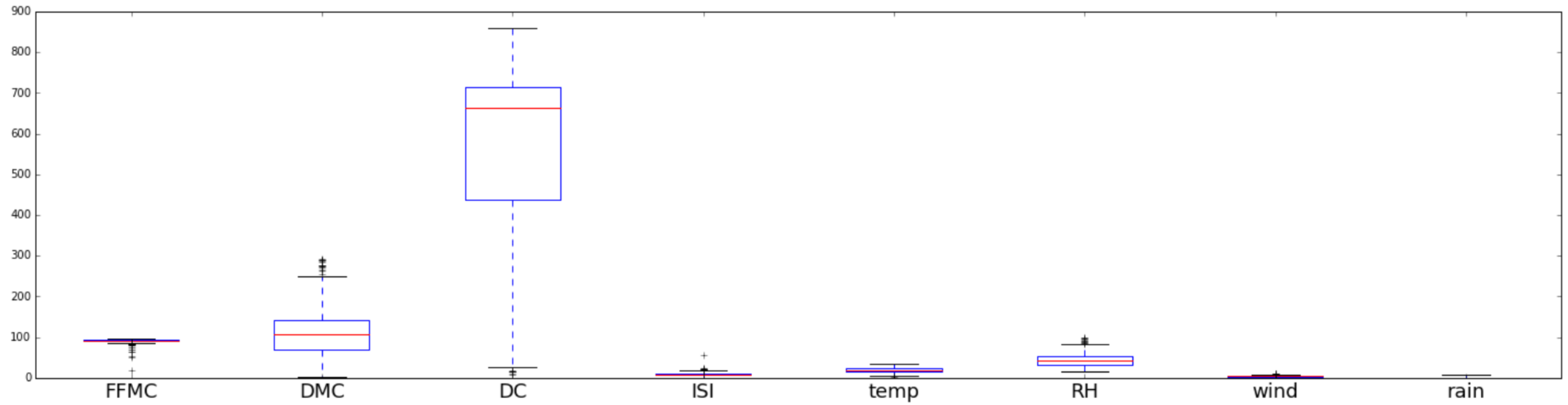
	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	month_may	month_nov	month_oct	month_sep	day_mon	day_sat	day_sun	day_thu
0	7	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	7	4	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	7	4	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

3 rows × 30 columns

Initial Analysis

Vizualizations - BoxPlots

```
plt.figure(figsize = (25,6))
plt.boxplot([data['FFMC'], data['DMC'],data['DC'],data['ISI'],
             data['temp'],data['RH'],data['wind'],data['rain']])

plt.xticks([1,2,3,4,5,6,7,8],['FFMC', 'DMC','DC','ISI','temp','RH','wind','rain'], fontsize = 18)
plt.show()
```

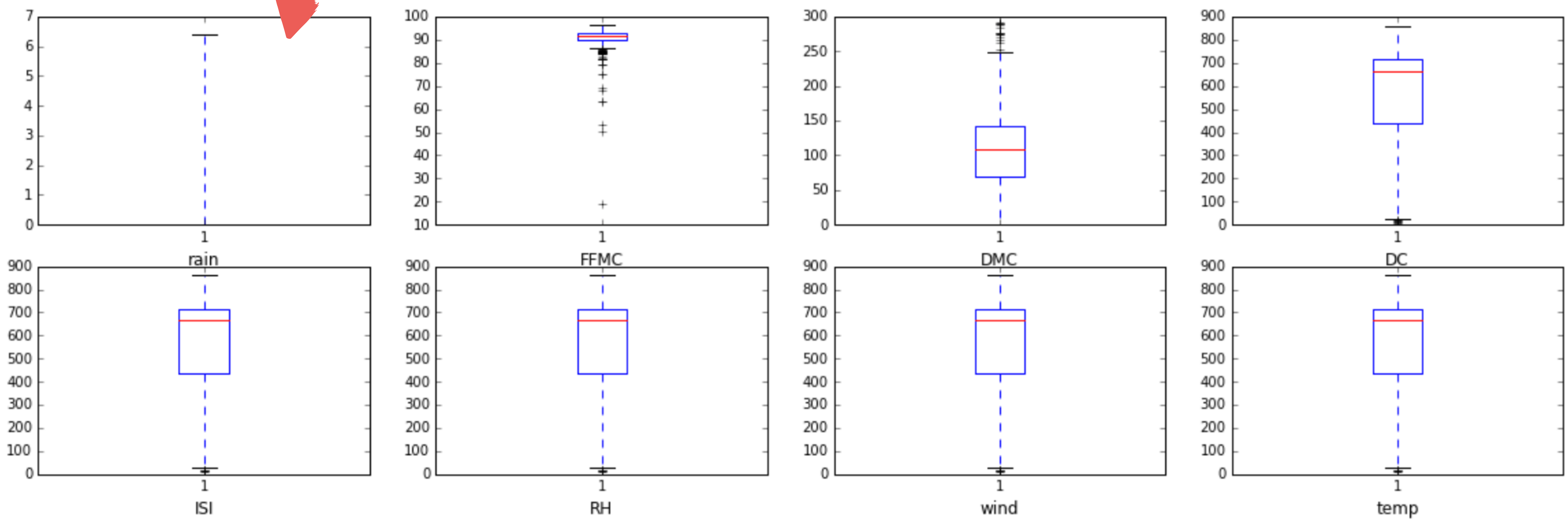


The range of values was too big to display all box plots together!

Initial Analysis

?

Separate box plots



Initial Analysis

```
data['rain'].value_counts() #Very dry place
```

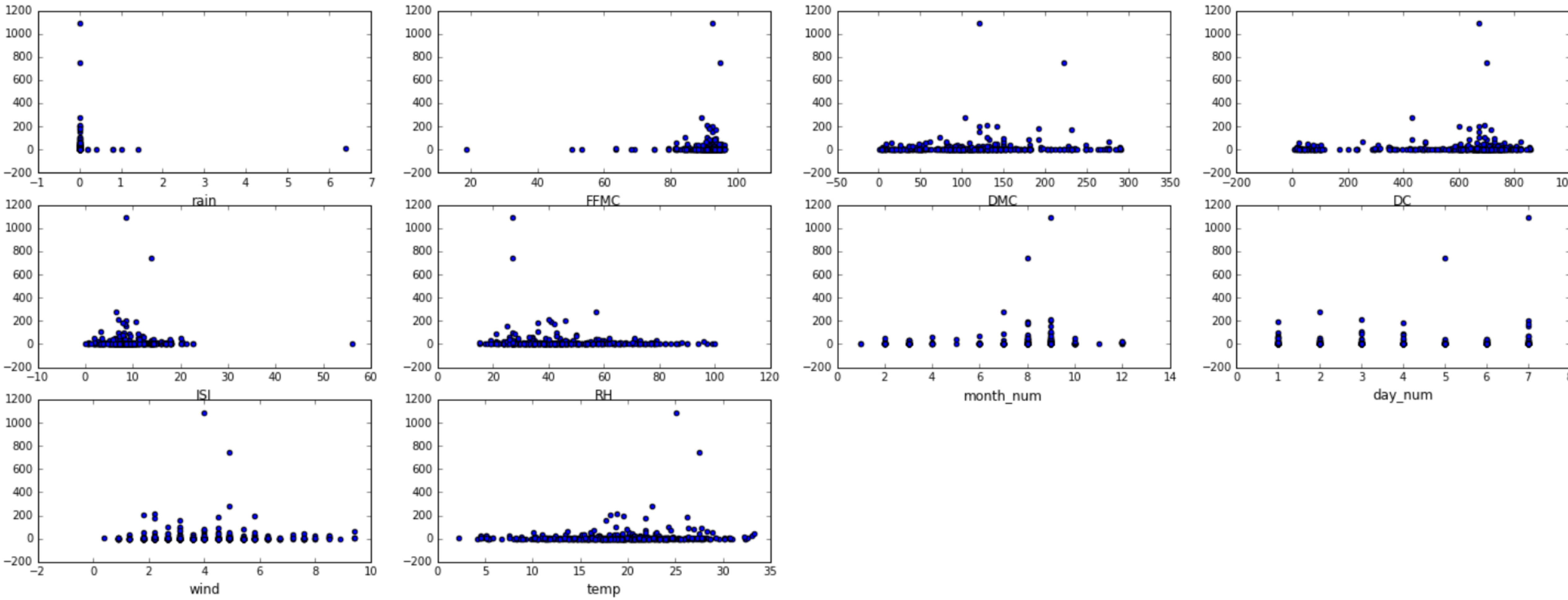
```
0.0    509
0.8      2
0.2      2
0.4      1
6.4      1
1.4      1
1.0      1
Name: rain, dtype: int64
```

The values for rain are in milliliters. → From 517 observations, 509 had no rain at all.

All models were used with and without "rain" and the results were approximately the same.

Initial Analysis

Scatter Plots



Initial Analysis

Defined X and Y

y for Regression

yclass for Classification

```
data["fire_yn"] = 0
data.fire_yn.loc[data['area'] > 0] = 1

ListOfAllVariables = data.columns.values
#print(ListOfAllVariables)
X = data[['FFMC','DMC','DC','ISI','temp','RH','wind','month_aug','month_dec', #9
          'month_feb', 'month_jan', 'month_jul', 'month_jun','month_mar', 'month_may', #6
          'month_nov', 'month_oct', 'month_sep','rain','day_mon', 'day_sat', 'day_sun', #7
          'day_thu', 'day_tue', 'day_wed']] #3
y = data['area']
yclass = data['fire_yn']
```

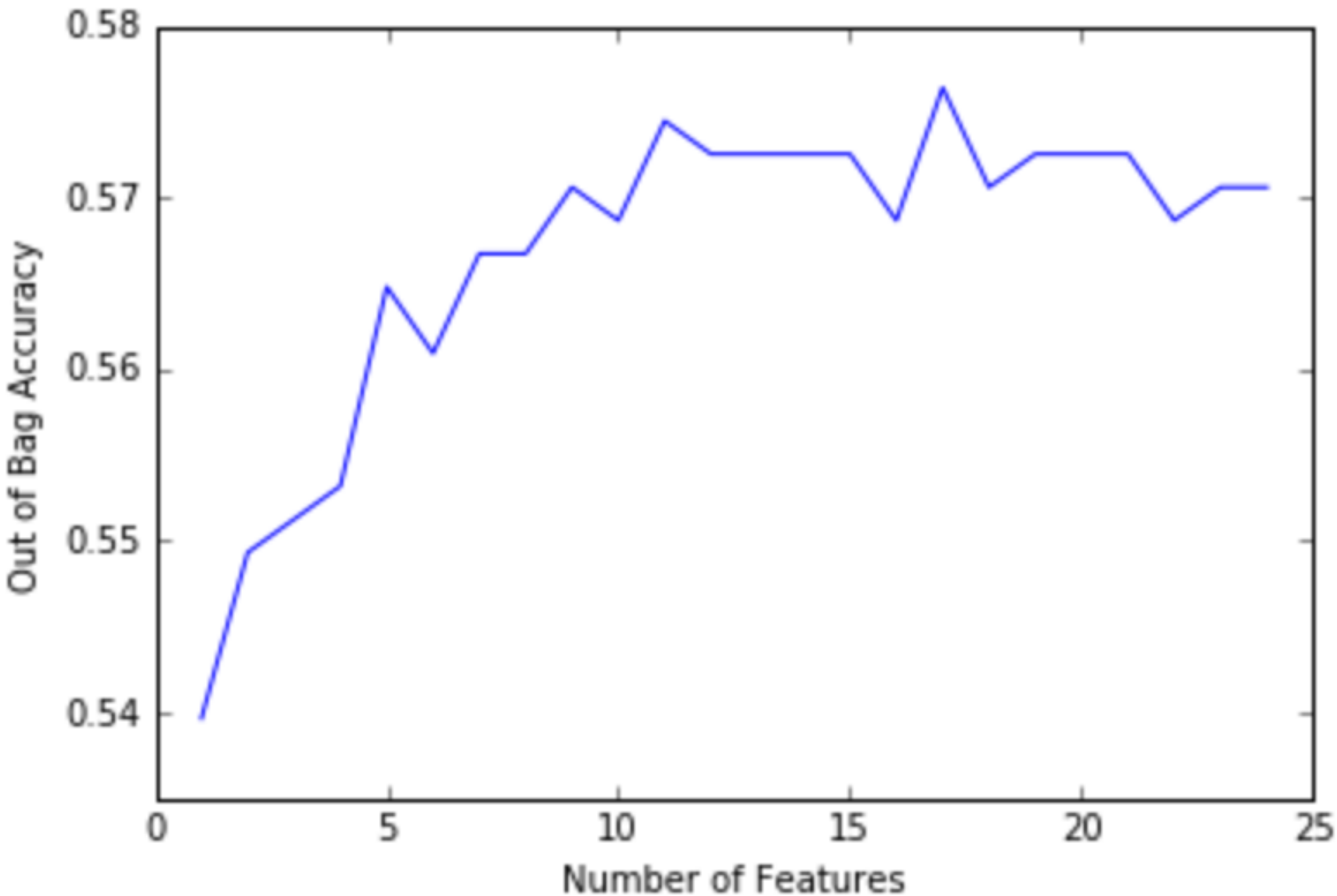
*The variable "rain" did not influenced the results.

Part II: Predicting a fire

Randon Forest Classifier

```
Features = range(1,25)
oob_score_RF = []
for i in Features:
        RFClass = RandomForestClassifier(n_estimators = 10000,
                                   max_features = i,
                                   min_samples_leaf = 5,
                                   oob_score = True,
                                   random_state = 1,
                                   n_jobs = -1)
        RFClass.fit(X,yclass)
        oob_score_RF.append(RFClass.oob_score_)

plt.plot(Features, oob_score_RF)
plt.xlabel("Number of Features")
plt.ylabel("Out of Bag Accuracy")
plt.show()

print("Out of Bag Accuracy  = %f" %RFClass.oob_score_)
scores = cross_val_score(RFClass, X, yclass, cv = 10)
print("Cross-validation Accuracy = %f" %scores.mean())
```

Classification




```
Out of Bag Accuracy  = 0.570600
Cross-validation Accuracy = 0.448831
```

Classification

Multiple + Voting

```
#clf1 = LogisticRegression()
clf2 = RandomForestClassifier(max_depth = 5, n_estimators = 10000)
clf3 = BernoulliNB()
clf4 = neighbors.KNeighborsClassifier(n_neighbors=199, weights='uniform')
clf5 = GaussianNB()
```

*Based on several attempts to run classification and regression models.

In both cases (voting = hard and soft), KNN seems to perform slightly better

Classification

Boosting Classification

```
NumberOfTrees = [100,1000,5000,10000,20000]
for i in NumberOfTrees:
    GBC_Tree = GradientBoostingClassifier(learning_rate = 0.01,
                                          n_estimators = i,
                                          max_depth = 2,
                                          min_samples_leaf = 10,
                                          random_state = 1)

kf = cross_validation.KFold(len(data), n_folds = 10, shuffle = True)

scores = []

for train_index, test_index in kf:
        GBC_Tree.fit(X.iloc[train_index], yclass.iloc[train_index])
        y_hat_test = GBC_Tree.predict(X.iloc[test_index])
        scores.append(float(sum(y_hat_test == yclass.iloc[test_index]))/len(y_hat_test))

Score_GBC_CV = np.mean(scores)

print(Score_GBC_CV)
```

```
0.560935143288
```

Classification

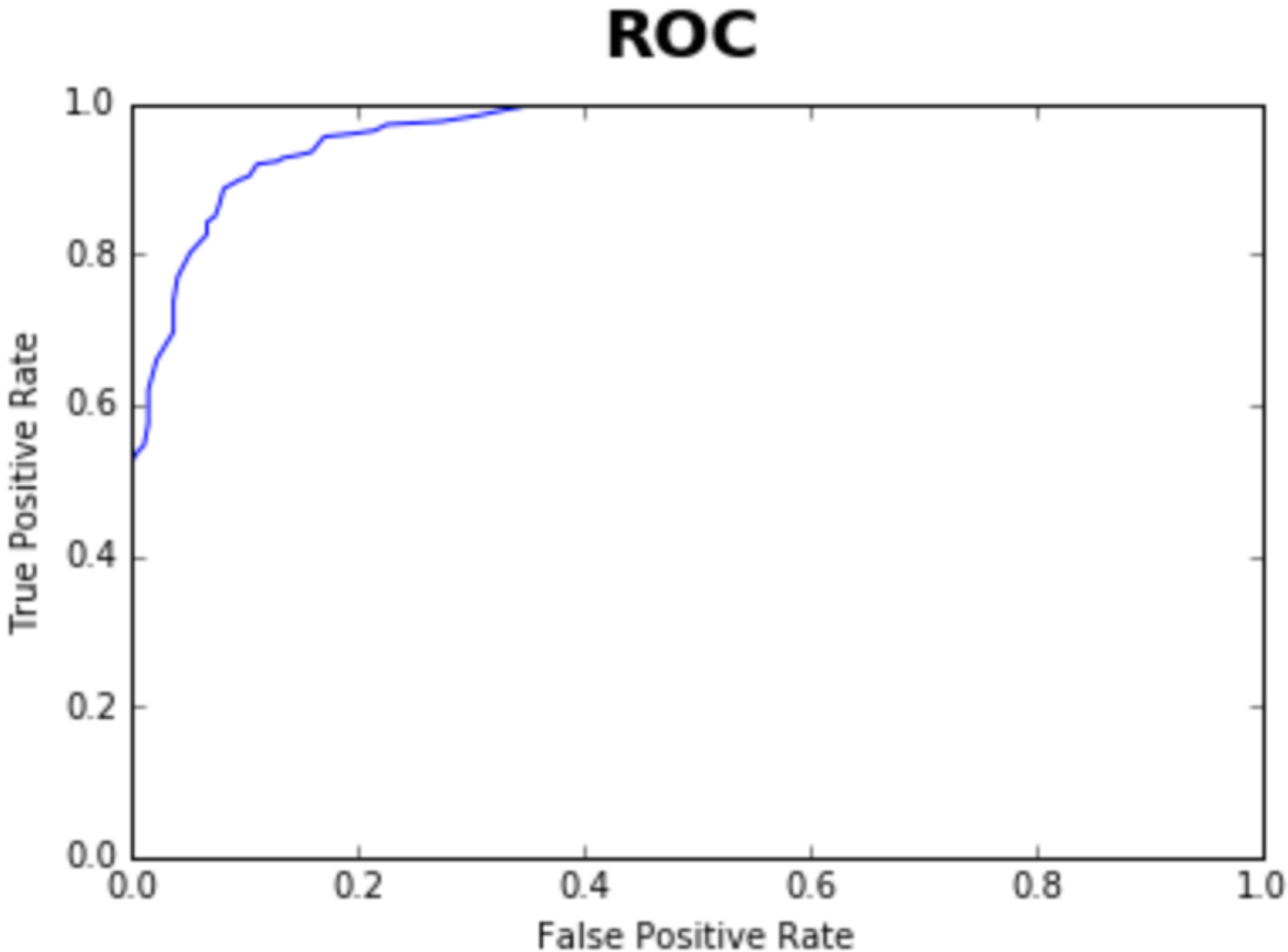
Confusion Matrix

```
#y_hat = RFClass.predict(X)
y_hat = GBC_Tree.predict(X)
confusion_matrix(yclass, y_hat)
```

```
array([[232,  15],
       [ 20, 250]])
```

```
False Positive Rate = 0.074074
False Negative Rate = 0.060729
True Positive Rate = 0.939271
True Negative Rate = 0.925926
Misclassification Error = 0.067698
Accuracy = 0.932302
```

Classification



Part III: Predicting the affected area

Regression

Park area = 75000 ha

Decision Trees → 65.62 ha

Random Forest → 64.32 ha

KNN → 64.53 ha

Boosting → 68.4 ha

68% of the time my prediction is within 68.4 ha from reality.
95% of the time my prediction is within 136.8 ha from reality.

Regression

```
(0.01980225268314029, 'month_aug'),
(0.025151129716881708, 'month_sep'),
(0.057454739710264456, 'day_sat'),
(0.077926134041150327, 'wind'),
(0.082412547758201449, 'ISI'),
(0.089339478901834218, 'DC'),
(0.1025578919855132, 'FFMC'),
(0.12466673555985694, 'DMC'),
(0.12607468830846508, 'RH'),
(0.16540980955026477, 'temp')]
```

Could suggest that some fires are caused by humans in holidays

```
(0.0, 'rain'),
```

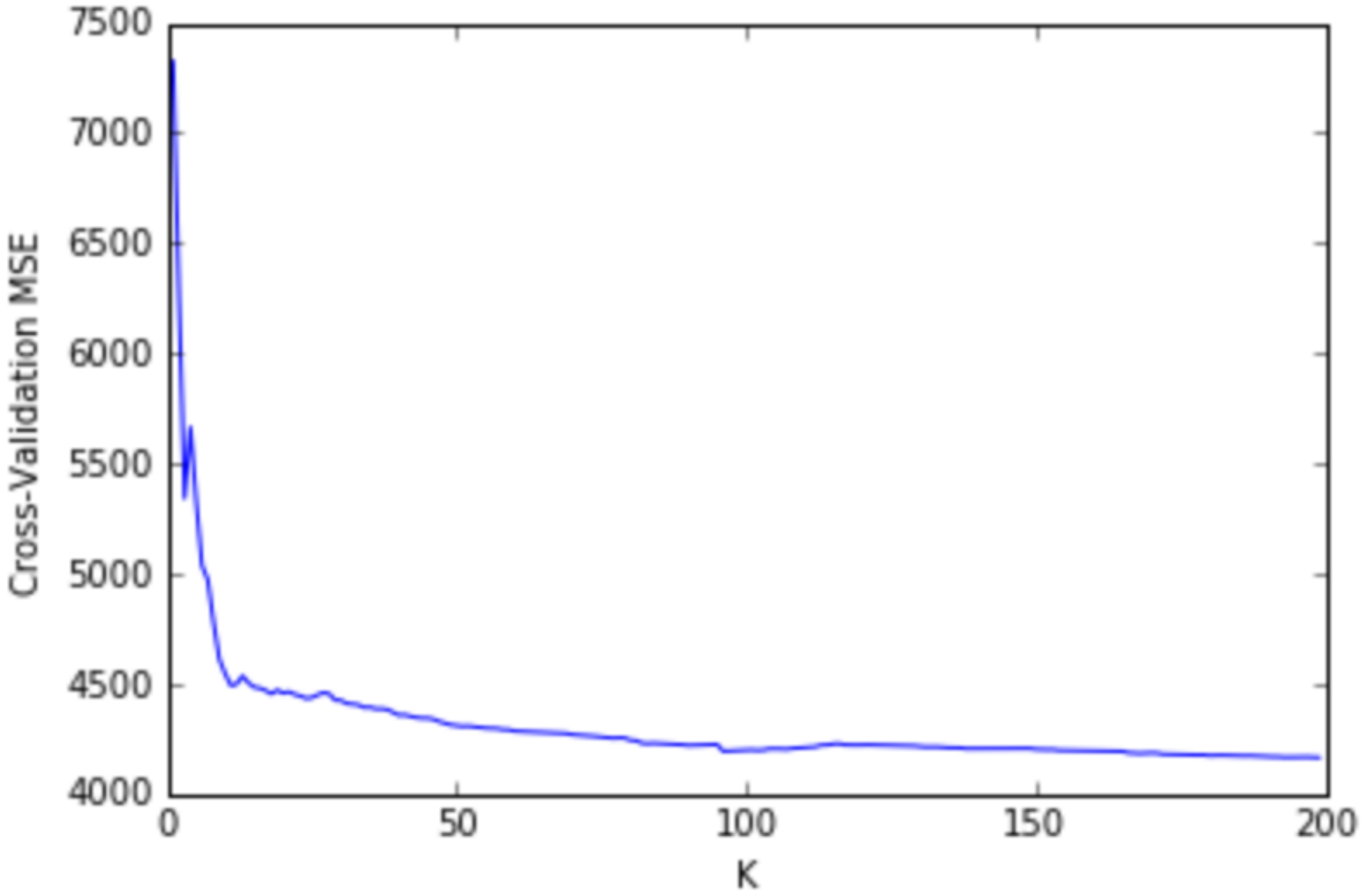
Regression

KNN

```
CV_Scores = []
RangeOfK = range(1,200)

for k in RangeOfK:
    knn = neighbors.KNeighborsRegressor(n_neighbors = k, weights = 'uniform')
    CV_Scores.append( -cross_val_score(knn, X, y, cv=10,
                                scoring = 'mean_squared_error').mean())


plt.plot(RangeOfK, CV_Scores)
plt.xlabel("K")
plt.ylabel("Cross-Validation MSE")
plt.show()

print "The best K is %f" %RangeOfK[np.argmin(CV_Scores)]
```

Regression

KNN



The best K is 199.000000

Conclusion

- The best classification model can predict with an accuracy of 93.2% if there will be a fire.

- On average, the regression models can precise the area that will most likely be affected by a fire with an error of 64 ha 68% of the time and with an error of 128 ha 95% of the time.

- Temperature is the most important factor, followed by relative humidity and moisture levels. There may be cases when humans were behind the incidents.

Thank you!

Sources

- Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

- [Cortez and Morais, 2007] P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimarães, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. Available at: [Web Link]

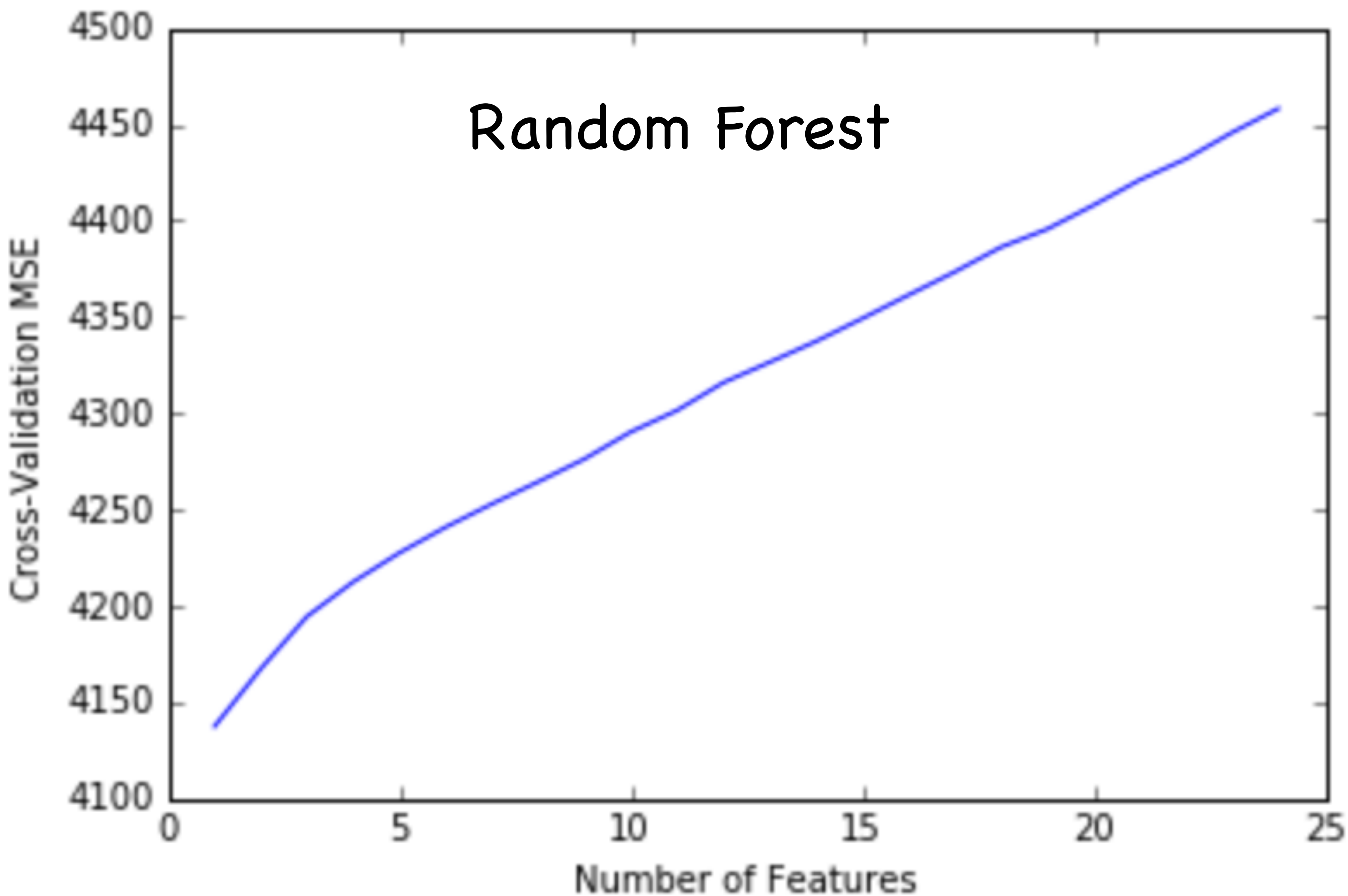
Appendix

Regression

Random Forest

```
Features = range(1,25)
score = []
for i in Features:
        RF = RandomForestRegressor(n_estimators = 10000,
                                   max_features = i,
                                   min_samples_leaf = 10,
                                   oob_score = True,
                                   random_state = 1,
                                   n_jobs = -1)
        score.append(-cross_val_score(RF, X, y, cv = 10,
                                      scoring = 'mean_squared_error',
                                      n_jobs = -1).mean())

plt.plot(Features, score)
plt.xlabel("Number of Features")
plt.ylabel("Cross-Validation MSE")
plt.show()
```

Regression



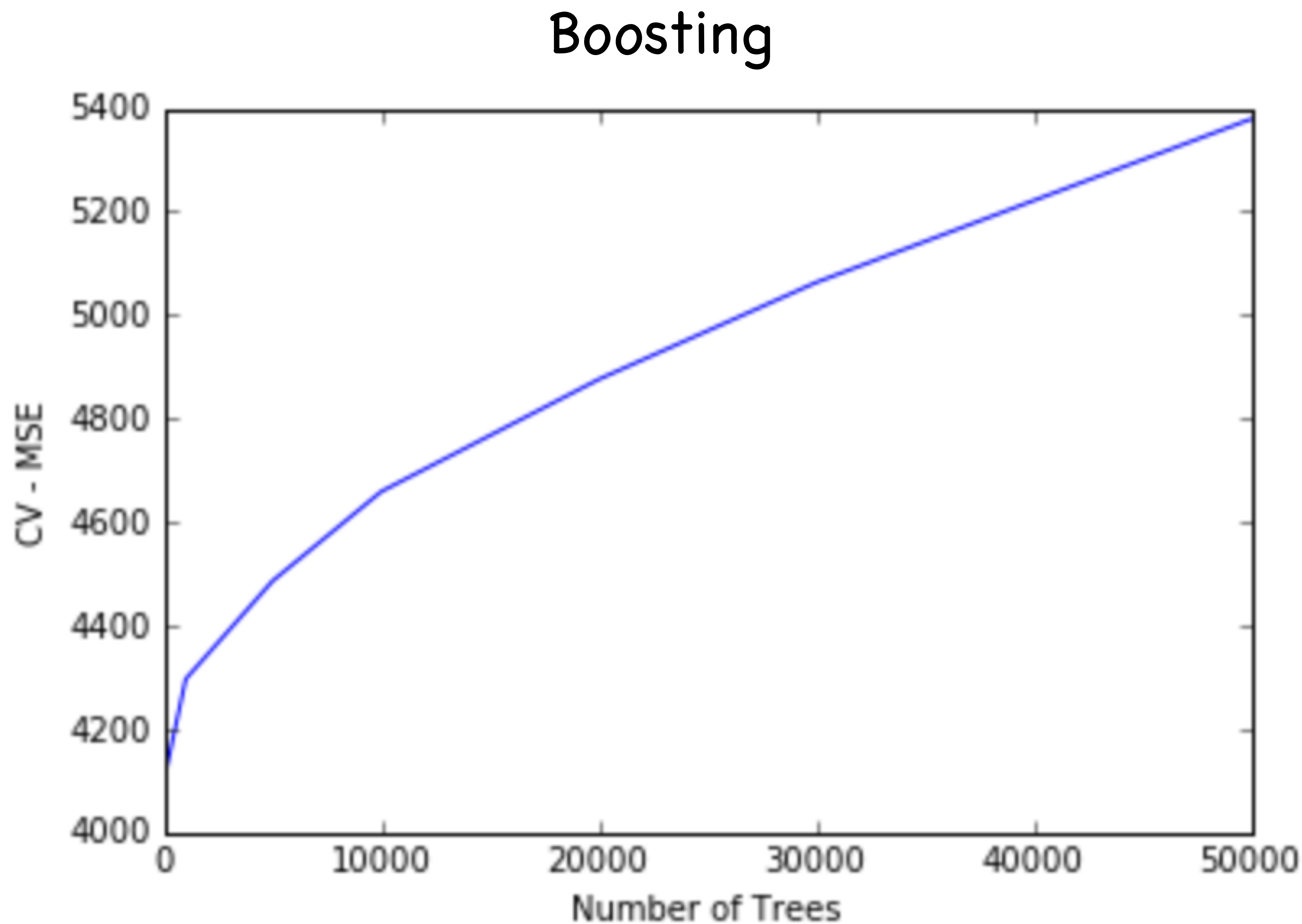
Optimal level of features is 1, which leads to a MSE of 4137.37

Regression

Boosting

```
Score = []
NumberOfTrees = [100,1000,5000,10000,20000,30000,40000,50000]
for i in NumberOfTrees:
                GBR_Tree = GradientBoostingRegressor(learning_rate = 0.01,
                                             n_estimators = i,
                                             max_depth = 2,
                                             min_samples_leaf = 10)

                Score.append(-cross_val_score(GBR_Tree, X, y, cv = 10,
                                              scoring = 'mean_squared_error', n_jobs = -1).mean())

plt.plot(NumberOfTrees, Score)
plt.xlabel("Number of Trees")
plt.ylabel("CV - MSE")
plt.show()
```

Regression



Boosting
5400
5200
5000
4800
4600
4400
4200
4000
CV - MSE
0
10000
20000
30000
40000
50000
Number of Trees