CS699 Data Mining

TERM PROJECT REPORT

Categorization of Property Type in Housing Website

Aditya Maheshwari
Vaidehi Shah

Dr. Jae Young Lee

BU MET, Spring 2023

INTRODUCTION

makaan.com is a renowned real estate portal based in India. It gives its customers various buying, selling and renting options for a wide range of properties all over India. This website has also implemented a rating system for brokers which helps the customers make a better informed decision.

The dataset used for this project has been prepared by scraping information off of makaan.com's website. The primary focus of this project is the listings that are stated as "sell" under the "Listing_Category" attribute. There are five distinct types of property breakdowns:

- Apartment
- Residential Plots
- Villas
- Independent Floor
- Independent House

The analysis of this project will be based on determining the type of property based on its features and characteristics. To avoid data imbalance, the categorization is done as "Apartments" and "Non-Apartments". By addressing this, the aim is to enhance the search engine of the Makaan.com website, providing more tailored search results that are in line with customers' preferences and requirements.

This project will implement five binary classification algorithms - Logistic Regression, K-Nearest Neighbors, Naïve Bayesian, Random Forest, Support Vector Machines - and five attribute selection methods - Chi-Square Test, Boruta Model, r-part, random forest feature importance, Recursive Feature Elimination. By implementing various classification algorithms and attribute selection methods, we will be able to understand which attributes are the strongest predictors of property_type.

PROBLEM STATEMENT

Precisely predicting the property type, whether it is an apartment or not, is a vital task within the real estate sector. This forecast can furnish valuable insights to investors, analysts, property managers, lenders, and homebuyers, facilitating market analysis, property management, investment decisions, and preference determination for potential buyers. A comprehensive analysis of a property's features and characteristics, such as its location, size, amenities, layout, building type, construction materials, and demographics of the surrounding area, can assist in making the prediction. By utilizing advanced analytical methods and machine learning algorithms, stakeholders can make informed decisions based on a wide range of factors, leading to maximizing returns and accomplishing their objectives.

MATERIALS AND METHODS

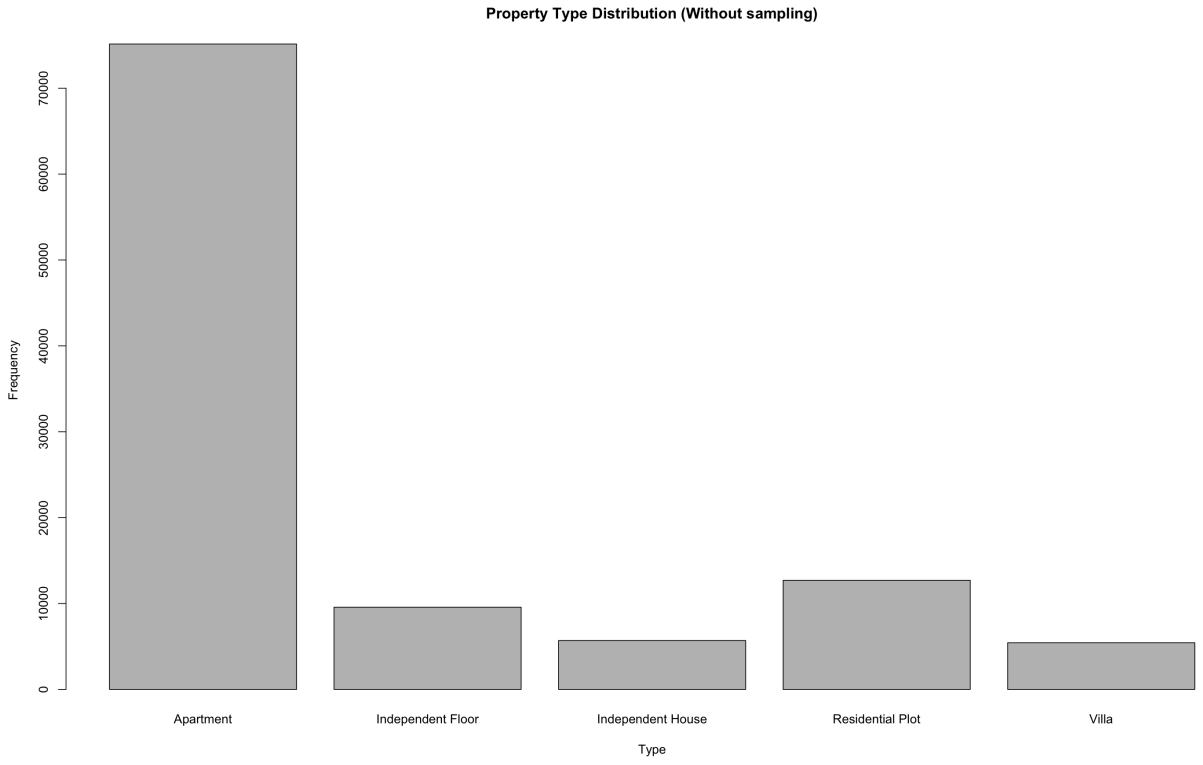
- *DATASET*

ATTRIBUTE NAME	DESCRIPTION	TARGET COLUMN ATTRIBUTES FOR CLASSIFICATION
Property_Name	Name of the property	
Property_id	ID of the property	
Property_type	Type of property	- Apartment - Non-Apartment (Independent floor, Independent house, residential plot, villa)
Property_status	Status of the property	Ready to Move, Under construction
Price_per_unit_area	Price per square feet in Indian Rupees	
Posted_On	Number of days since the Ad for the property was posted	
Project_URL	Link to the Ad of the property on Maakan.com	
builder_id	ID of the builder	
Builder_name	Name of the builder	
Property_building_status	Building status	Active, Inactive, Unverified
City_id	ID of the city	
City_name	City in which property is located	Ahemdabad, Bangalore, Chennai, Delhi, Hyderabad, Kolkata, Lucknow, Mumbai
No_of_BHK	Number of Bedrooms in the property	
Locality_ID	ID of the locality	
Locality_Name	Locality in which property is located	
Longitude	This specifies the exact location of the property	
Latitude		
Price	Price of the property in Indian rupees	
Size	Size of property in terms of sq ft	
Sub_urban_ID	ID of the sub urban area	
Sub_urban_name	Name of the sub urban area	
description	Description of the property	
is_furnished	If the property is furnished or not	Furnished, Unfurnished, Semi-furnished
listing_domain_score	A domain score in real estate is a rating system used by the real estate website "Domain" to rate a property based on its characteristics, such as location, size, and features.	
is_plot	If the property is a plot	TRUE, FALSE

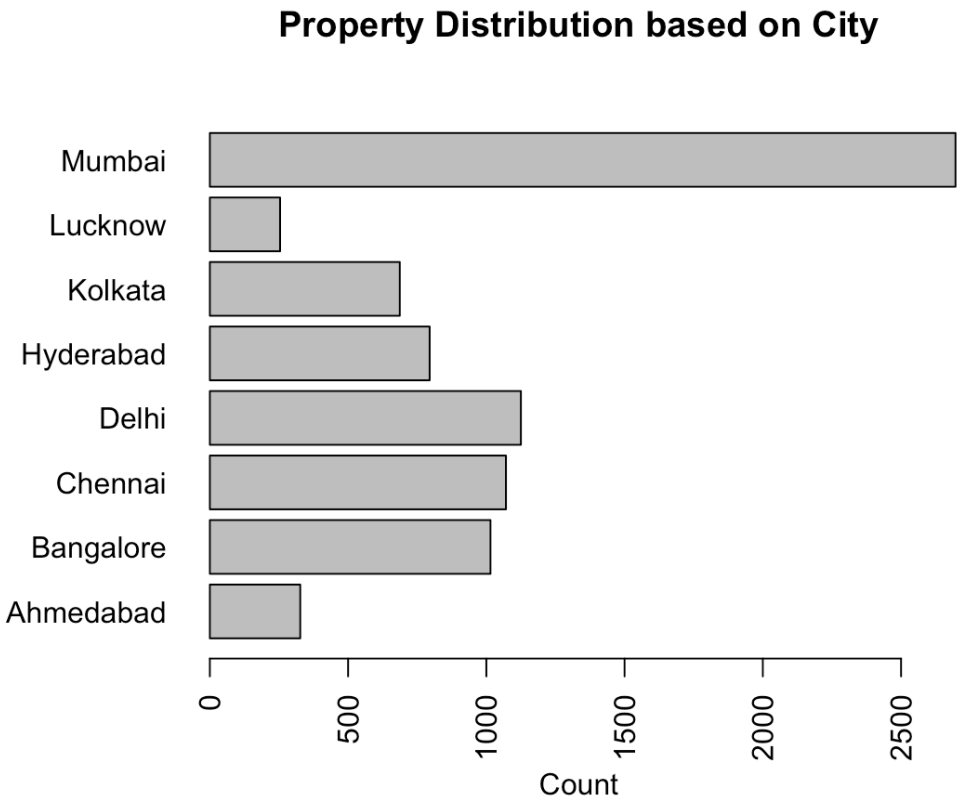
ATTRIBUTE NAME	DESCRIPTION	TARGET COLUMN ATTRIBUTES FOR CLASSIFICATION
is_RERA_registered	If the propperty is registered with Real Estate Regulatory Authority of India	TRUE, FALSE
is_Apartment	If the property is an Apartment	TRUE, FALSE
is_ready_to_move	Is the property is ready to move in	TRUE, FALSE
is_commercial_Listing	Is the property a commericial listing	TRUE, FALSE
is_PentaHouse	Is the property a penthouse or studio	TRUE, FALSE
is_studio		TRUE, FALSE
Listing_Category	Listing Category of the property on Maakan.com	

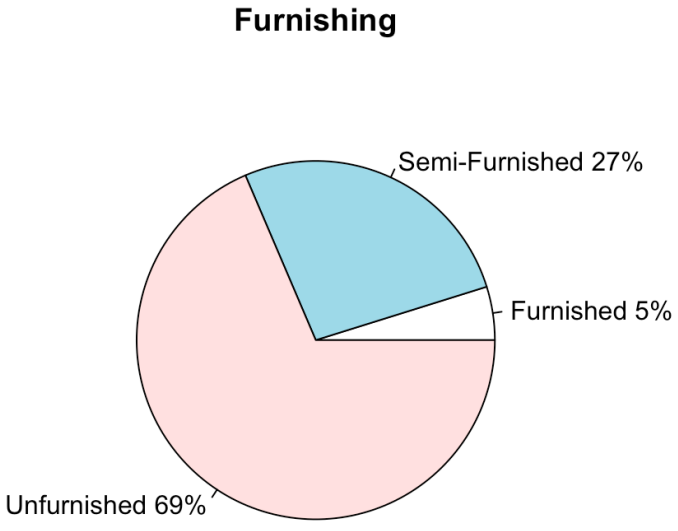
- *DATA VISUALIZATION*

During our Exploratory Data Analysis, we employed visualizations to enhance our understanding of the data, particularly the various categorical variables and our target variable. We mapped various categorical variables and our target variable to discern patterns and insights. We observed that the dataset exhibited some bias in relation to our target variable. To address this issue, we undertook preprocessing measures and worked on the target variable to ensure that our dataset was balanced, which is vital for obtaining reliable insights. These steps were necessary to optimize our data for further analysis, enabling us to derive meaningful and accurate conclusions.

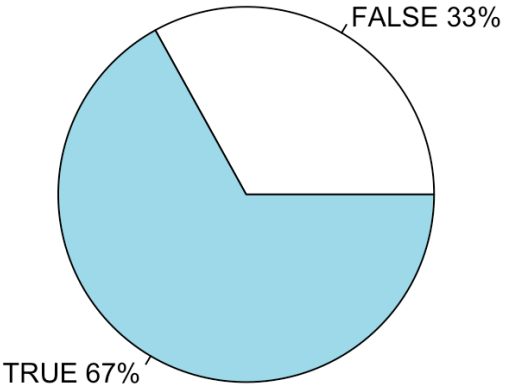


```
> table(df$Property_type)

    Apartment Non-Apartment
        75147         33403
> prop.table(table(df$Property_type))

    Apartment Non-Apartment
    0.6922801     0.3077199
```

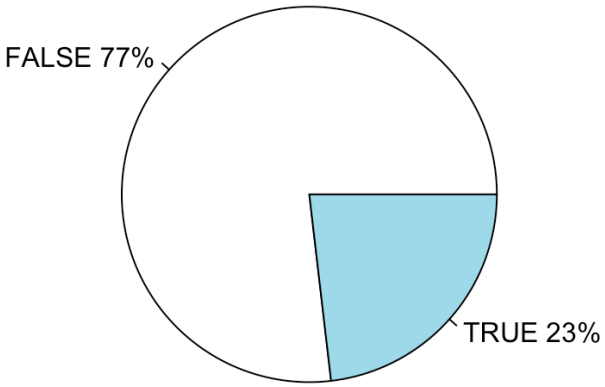




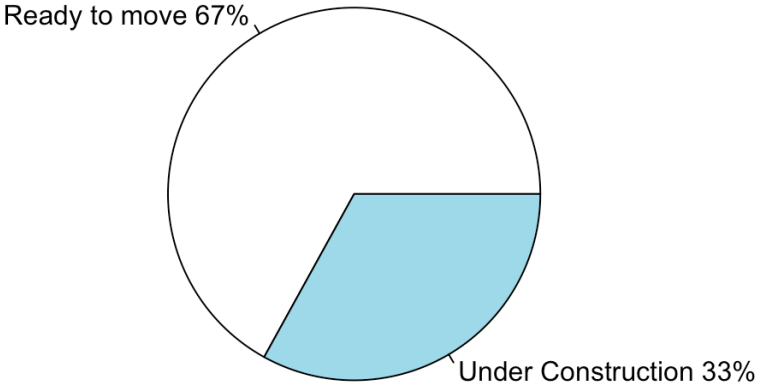
Ready to Move



RERA Registered



Property Status



- *DATA CLEANING*

For any sort of data analysis, a clean data set is necessary. The four steps below outline the process:

1. Remove Duplicate or Irrelevant Data:
   Certain attributes that are not relevant to the analysis exist in the dataset. The type of property is one-hot encoded which will hinder out analysis of the final target variable and so those columns have been removed. Moreover, attributes like the ID for areas/localities and property and builders have been identified as not necessary for the analysis. We got rid of such data.

2. Handle Missing Data:
   Most of the NA values in our dataset were in the columns builder ID and builder name. Out data was kind of biased. After getting rid of the NA values by simply removing those rows from the dataset, the data became a bit unbiased. We did not have to replace the data with mean or median because most of the data that was missing was categorical.

3. Not Removed Unwanted Outliers:
   For the analysis we are doing, depending on the size and location of the property, the prices would definitely have an impact on the analysis. We believe these outliers for prices and sizes of the house belong to the distribution and decided to keep the outlier values. As part of the preprocessing steps, these numeric variables will be standardized, adjusting the extreme values in these attributes and analysis will be performed on that.

- *DATA TRANSFORMATION*

Data transformation was a very important part of our data preprocessing. We have a mix of categorical and numerical data which needed to be formatted in a way suitable for analysis to be performed on it. In order to do so, we needed to understand our data better. We did that by using the `str()` and `table()` functions. We got to know the distribution of our data and we set to data transformation.

1. Remove Unnecessary Columns:
   The `dplyr::select()` function is used to drop columns that are not required for the analysis. This step helps to reduce the size of the data frame and remove unnecessary information that may not be relevant for the analysis.

2. Data Type Transformation:
   Most of the numerical variables in our dataset were set to data type character, and not integer or double. We converted price, area and size of the property attributes to numeric. This was important to ensure that numeric variable are not used as text variables, but as numeric ones. The `factor()` function is used to convert categorical variables into factors. We had to convert property status, city name, locality name, property building status, furnished and ready to move to factors so that they did not hinder the performance of a model.

3. Min-Max Scaling:
   The `minMax()` function is defined to implement min-max scaling, which is a normalization technique that scales variables to a range of 0 and 1. This step is important to ensure that variables are on the same scale and to avoid any bias that may arise from variables that are measured on different scales. It is used to normalize the Price_per_unit_area, Price, and Size variables.

4. Target variable set up:
   For removing the bias from the data and predicting the target variable correctly, we categorized the target variable to be either Apartment or non-Apartment. We included categories other than “Apartment” under the umbrella of “non-Apartments” and we will be performing our analysis to predict if our property type is an apartment or not.

- *CLASSIFICATION ALGORITHMS*

Classification models are a type of machine learning algorithm used to predict the class or category of a data point based on its features or attributes. These models are trained on a labeled dataset, meaning that each data point is assigned to a known class. This project will use the following classification algorithms:

1. Logistic Regression:
   Logistic Regression (LR) is a type of supervised learning algorithm used for classification tasks. The algorithm works by calculating the probability of the input variables belonging to one of the two classes, using a logistic function and maps any input value to a value between 0 and 1. The algorithm then applies a decision threshold to these probabilities to make a final prediction.

2. kNN:
   The K-Nearest Neighbors (kNN) classification model categorizes data by calculating the distance between a test instance and all points in the training dataset. This process can be slow, but it helps identify the "k" number of training points that are closest to the test instance. Once the nearest neighbors are identified, the kNN algorithm calculates the probability of the test instance belonging to each class represented in the k-training data. The class with the highest probability is selected as the predicted category for the test instance. To optimize the performance of the kNN algorithm, the elbow method is often used.

3. Support Machine Vector:
   The Support Vector Machines (SVM) classification model works by constructing a hyperplane in a multidimensional space to separate the classes. SVM generates the optimal hyperplane in an iterative manner. This is done by minimizing the error and maximizing the margin between the support vectors and the dataset. The goal of the algorithm is to find a maximum marginal hyperplane that best divides the classes. s

4. Naive-Bayes:
   The Naïve Bayesian (NB) classification model is a statistical classifier that uses probability to predict labels. This model is known for its simplicity and ease of implementation. It works by implementing Bayes' Theorem of conditional probability to calculate the likelihood of each class label, given the data point belongs to a particular class. The predicted class is the one with the highest probability.

5. Random Forest:
   The Random Forest (RF) classification model is an ensemble learning algorithm that consists of multiple decision trees. The model operates by randomly selecting a subset of samples from the dataset and constructing a decision tree for each sample. The decision trees then predict the classification value for each sample, and a vote is performed for each predicted result. The prediction with the highest number of votes is selected as the final prediction. This approach results in a more accurate and robust model, reducing the risk of overfitting to the training data.

- *ATTRIBUTE SELECTION METHODS*

Attribute selection is a process in machine learning that involves selecting the most relevant and useful features (or attributes) from a dataset to use as input for a model. This process is crucial for optimizing the model's accuracy and reducing its complexity, thereby improving its performance. We will be using the following attribute selections methods for the project:

1. Chi-Square Test:
   The Chi-Square attribute selection technique employs the Chi-Square test to evaluate the degree of independence between two occurrences. It applies the Chi-Square test individually to each attribute or predictor variable and the class attribute or response variable. When the Chi-Square test shows that the predictor variable and the class variable are independent, then the predictor variable is excluded. Otherwise, it is regarded as an essential predictor and retained for the classification model.

2. Boruta Model:
   The Boruta Model operates by using the random forest method to assess the importance of each feature. In the training phase, the Boruta Model generates a set of random shadow features and adds them to the original dataset. The Mean Decrease Impurity (MDI) method is used to evaluate the importance of each feature. The Boruta Model repeats this process iteratively until it can distinguish between the truly important features and the less important ones. The Boruta Model is particularly effective in handling datasets that are high dimensional and noisy.

3. R-part:
   Recursive partitioning operates by repeatedly partitioning the data into subsets, evaluating all possible feature splits at each node, and selecting the split that optimizes a particular criterion, such as information gain or Gini impurity. Features that are selected at the top of the tree are deemed to be more significant predictors of the target variable, whereas those appearing further down are considered to be less important. One drawback of rpart is that it can be susceptible to overfitting.

4. Random Forest Feature Importance:
   During Random Forest training, each decision tree is created using a random subset of features selected from the entire feature set. By amalgamating these trees, Random Forest can develop a strong and resilient model. To assess the significance of features in the dataset, Random Forest estimates the importance of each feature based on its contribution to the model's accuracy. This evaluation involves calculating a feature importance score that is then normalized between 0 and 1, so that the sum of all feature importance scores is equivalent to 1.

5. Recursive Feature Elimination:
   Recursive Feature Elimination (RFE) is a feature selection technique used to identify the most important subset of features in a dataset. It does so by iteratively eliminating the least important features in a model until a desired number of features is achieved. RFE can reduce the dataset's dimensionality, improve model performance, and provide insights into which features are most significant for predicting the target variable.

- *DATA MINING TOOLS & PROCEDURES*

In this project, we employed R to carry out different tasks such as data preparation, attribute selection, implementation of classification algorithms, and result evaluation. To perform these tasks, we used various R packages including caret, rsample, FSelector, and pROC. The caret package was useful in standardizing and centering the data, implementing classification algorithms such as Naive-Bayes, kNN, Logistic Regression, Support Machine Vector, and Random Forest, and evaluating the models using metrics such as accuracy, confusion matrix, precision, recall, and F1-measure. The rsample package helped us with tasks such as cross-validation and train/test split. We also utilized the FSelector package to perform attribute selection using methods such as Chi-Square, Boruta Model, R-part, Random Forest Feature Importance, Recursive Feature Elimination. Finally, we used the pROC package to perform ROC AUC analysis.

Through the use of R and its extensive range of packages for data preparation, analysis, modeling, and visualization, we were able to benefit from the flexibility and independence offered by these tools during the knowledge discovery process. Our project is contained within RStudio, which allows us to organize the R code with text cells for documentation and comments. The code is divided into three main sections: Preparation, Analysis, and Evaluation.
The Preparation section is responsible for executing the code and performs the following tasks: 1) loading relevant libraries and packages, 2) reading the initial dataset into a Pandas data frame, 3) cleaning the data using a four-step procedure, 4) exploring the data through visualization methods, and 5) implementing the necessary preprocessing steps for the following sections.

In the Analysis section of the R code, the first step involves defining the classification algorithms that will be used: Naive-Bayes, kNN, Logistic Regression, Support Machine Vector, and Random Forest. The next step involves implementing 10-fold cross-validation on the entire dataset using these five classification algorithms. The average metrics for accuracy, true positive rate, false positive rate, precision, recall, F1-measure, Matthews correlation coefficient (MCC), and ROC AUC for the 10 folds were then collected to evaluate the effectiveness of the classification algorithms.

After splitting the dataset into a training set (66%) and a test set (34%), the five attribute selection methods: Chi-Square, Boruta Model, R-part, Random Forest Feature Importance, and Recursive Feature Elimination were implemented. For each of the five classification algorithms, the attributes selected by each method were used as predictor variables. To evaluate the effectiveness of the attribute selection methods and classification algorithms, the Accuracy, Confusion Matrix, True Positive Rate, False Positive Rate, Precision, Recall, F1-Measure, MCC, and ROC AUC were collected for a total of 30 models. Finally, the findings were summarized and visualized in the Evaluation section of the R Code to determine the overall best model

RESULTS

Attributes Selected:
Attribute selection methods were employed to minimize the dimensionality of the dataset in hopes of increasing the accuracy and efficiency of the classification algorithms. We are interested in which attributes were selected by which attribute selection methods. It will be interesting to understand if the classification algorithm's performance increases or decreases with more or less predictor variables.

FEATURES SELECTION METHOD	NUMBER OF SELECTED FEATURES
Random Forest Feature Selection	4
Boruta Model	12
Chi-Square	5
R-Part	10
Recursive Feature Elimination	5

ALL FEATURES

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	89.7%
Naive Bayes	79.3%
k-NN	93.94%
Support Vector Machine	90.85%
Random Forest	95.69%

CLASSIFIER 1: GLM Model (Logistic regression)

```
> log.model
Generalized Linear Model

71665 samples
   12 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64498, 64498, 64498, 64499, 64499, 64498, ...
Resampling results:

  Accuracy   Kappa
  0.8950116  0.7475553

> #summary(log.model)
> log.pred <- predict(log.model, newdata = test)
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> log.cm <- table(test$Property_type, log.pred)
> print(log.cm)
               log.pred
                Apartment Non-Apartment
  Apartment         24191          1322
  Non-Apartment      2474          8931
> log.results <- confusionMatrix(log.cm)
> log.acc <- log.results$overall['Accuracy']
> print(log.acc)
 Accuracy
0.8971775
> log.metrics <- binary_metrics(test$Property_type, log.pred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9481833 0.21692240 0.9072192 0.9481833 0.9272490 0.8656304 0.7544049
2 Non-Apartment 0.7830776 0.05181672 0.8710621 0.7830776 0.8247299 0.8656304 0.7544049
3  weighted_avg 0.8656304 0.13436956 0.8891407 0.8656304 0.8759895 0.8656304 0.7544049
`
```

CLASSIFIER 2: Naive Bayes Model

```
> n.model
Naive Bayes

71665 samples
   12 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64498, 64499, 64498, 64498, 64499, 64499, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7480779  0.2741092
   TRUE      0.7876934  0.3954673

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
> #summary(n.model)
> n.pred <- predict(n.model, newdata = test)
> n.cm <- table(test$Property_type, n.pred)
> n.results <- confusionMatrix(n.cm)
> n.acc <- n.results$overall['Accuracy']
> print(n.acc)
 Accuracy
0.7934612
> n.metrics <- binary_metrics(test$Property_type, n.pred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics)
          class   TP_rate     FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9914945 0.649539676 0.7734834 0.9914945 0.8690245 0.6709774 0.4968777
2 Non-Apartment 0.3504603 0.008505468 0.9485050 0.3504603 0.5118125 0.6709774 0.4968777
3  weighted_avg 0.6709774 0.329022572 0.8609942 0.6709774 0.6904185 0.6709774 0.4968777
```

CLASSIFIER 3: K-Nearest Neighbors

```
> knn.model
k-Nearest Neighbors

71665 samples
   12 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64498, 64499, 64498, 64499, 64498, 64499, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9376544  0.8532263
   2  0.9326031  0.8410944
   3  0.9388544  0.8552590
   4  0.9362869  0.8491339
   5  0.9375148  0.8518825
   6  0.9342775  0.8441995
   7  0.9344171  0.8443238
   8  0.9329659  0.8409720
   9  0.9328403  0.8405068
  10  0.9316961  0.8377744
  11  0.9310682  0.8361687
  12  0.9301193  0.8338600
  13  0.9298402  0.8331835
  14  0.9291984  0.8316226
  15  0.9291565  0.8314677

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 3.
> knn.pred <- predict(knn.model, newdata = test)
> knn.cm <- table(test$Property_type, knn.pred)
> knn.cm
               knn.pred
                Apartment Non-Apartment
  Apartment         24629           884
  Non-Apartment      1351         10054
```

```
> knn.results <- confusionMatrix(knn.cm)
> knn.results
Confusion Matrix and Statistics

               knn.pred
                Apartment Non-Apartment
  Apartment         24629           884
  Non-Apartment      1351         10054

               Accuracy : 0.9395
                 95% CI : (0.937, 0.9419)
    No Information Rate : 0.7037
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8566

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9480
            Specificity : 0.9192
         Pos Pred Value : 0.9654
         Neg Pred Value : 0.8815
             Prevalence : 0.7037
         Detection Rate : 0.6671
   Detection Prevalence : 0.6911
      Balanced Accuracy : 0.9336

       'Positive' Class : Apartment

> knn.acc <- knn.results$overall['Accuracy']
> print(knn.acc)
 Accuracy
0.9394604
> knn.metrics <- binary_metrics(test$Property_type, knn.pred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9653510 0.11845682 0.9479985 0.9653510 0.9565960 0.9234471 0.8569767
2 Non-Apartment 0.8815432 0.03464900 0.9191808 0.8815432 0.8999687 0.9234471 0.8569767
3  weighted_avg 0.9234471 0.07655291 0.9335896 0.9234471 0.9282824 0.9234471 0.8569767
```

CLASSIFIER 4: Support Vector Machine

```
> svm.results
Confusion Matrix and Statistics

               svm.pred
                Apartment Non-Apartment
  Apartment         24243          1270
  Non-Apartment      2105          9300

               Accuracy : 0.9086
                 95% CI : (0.9056, 0.9115)
    No Information Rate : 0.7137
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7815

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9201
            Specificity : 0.8798
         Pos Pred Value : 0.9502
         Neg Pred Value : 0.8154
             Prevalence : 0.7137
         Detection Rate : 0.6567
   Detection Prevalence : 0.6911
      Balanced Accuracy : 0.9000

       'Positive' Class : Apartment

> svm.acc <- svm.results$overall['Accuracy']
> print(svm.acc)
 Accuracy
0.9085812
> svm.metrics <- binary_metrics(test$Property_type, svm.pred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9502215 0.18456817 0.9201078 0.9502215 0.9349222 0.8828266 0.7826169
2 Non-Apartment 0.8154318 0.04977854 0.8798486 0.8154318 0.8464164 0.8828266 0.7826169
3  weighted_avg 0.8828266 0.11717336 0.8999782 0.8828266 0.8906693 0.8828266 0.7826169
```

CLASSIFIER 5: Random Forest

```
> rf.pred <- predict(rf.model, newdata = test)
> rf.cm <- table(test$Property_type, rf.pred)
> # Confusion Matrix
> rf.results <- confusionMatrix(rf.cm)
> rf.results
Confusion Matrix and Statistics

               rf.pred
                Apartment Non-Apartment
  Apartment         24894           656
  Non-Apartment       932         10425

               Accuracy : 0.957
                 95% CI : (0.9549, 0.959)
    No Information Rate : 0.6998
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8983

 Mcnemar's Test P-Value : 5.166e-12

            Sensitivity : 0.9639
            Specificity : 0.9408
         Pos Pred Value : 0.9743
         Neg Pred Value : 0.9179
             Prevalence : 0.6998
         Detection Rate : 0.6745
   Detection Prevalence : 0.6923
      Balanced Accuracy : 0.9524

       'Positive' Class : Apartment

> # Accuracy
> rf.acc <- rf.results$overall['Accuracy']
> print(rf.acc)
 Accuracy
0.9569729
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted averc
 measures
> rf.metrics <- binary_metrics(test$Property_type, rf.pred)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9743249 0.08206393 0.9639123 0.9743249 0.9690906 0.9461305 0.8984648
2 Non-Apartment 0.9179361 0.02567515 0.9407996 0.9179361 0.9292272 0.9461305 0.8984648
3  weighted_avg 0.9461305 0.05386954 0.9523560 0.9461305 0.9491589 0.9461305 0.8984648
```

RANDOM FOREST FEATURE SELECTION

```
> ############################## CLASSIFIERS (RANDOM FOREST FEATURE SELECTION) ##############################
> # Build random forest model
> rf_model <- randomForest(Property_type ~ ., data = train, importance = TRUE)
> rf_model

Call:
 randomForest(formula = Property_type ~ ., data = train, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 4.55%
Confusion matrix:
              Apartment Non-Apartment class.error
Apartment         48272          1368  0.02755842
Non-Apartment      1895         20246  0.08558782
```

```
> selected_features <- rownames(importance_scores)[order(importance_scores[,1], decreasing =TRUE)][1:4]
> selected_features
[1] "No_of_BHK"           "Size"                "Price_per_unit_area" "Price"
```

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	83.63%
Naive Bayes	79.25%
k-NN	85.72%
Support Vector Machine	84.73%
Random Forest	87.05%

CLASSIFIER 1: GLM Model (Logistic regression)

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24960	612
Non-Apartment	5441	5965

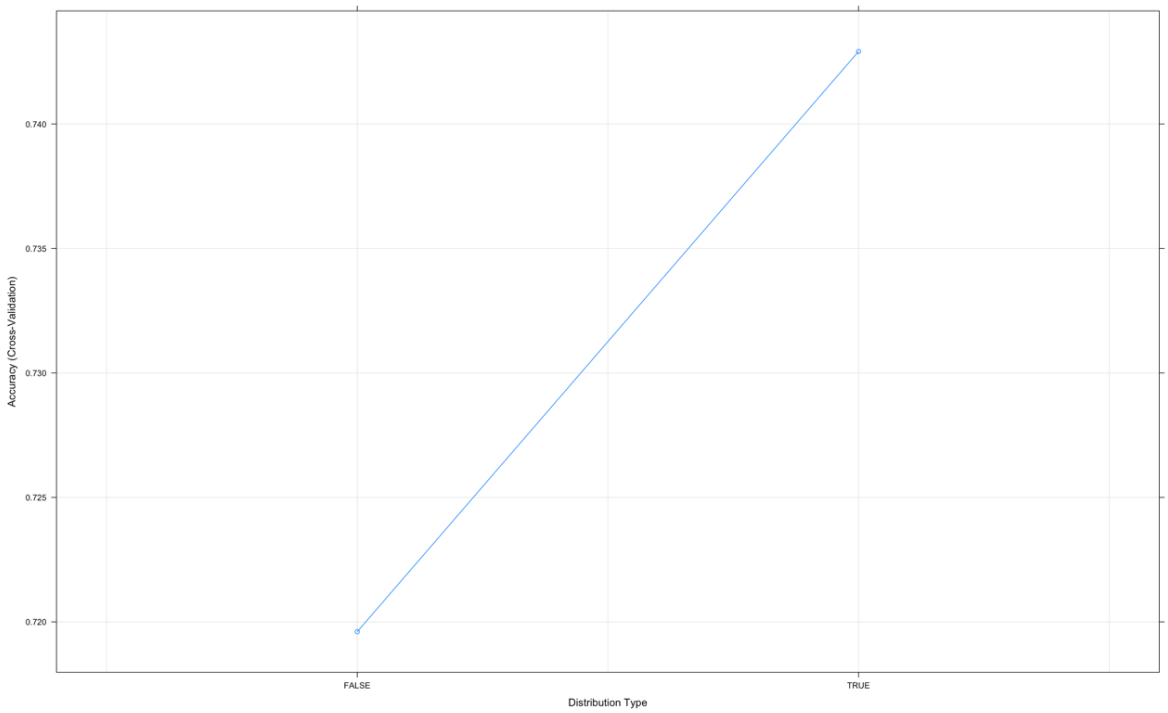
```
> log.model1
Generalized Linear Model

71781 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64602, 64603, 64603, 64603, ...
Resampling results:

  Accuracy   Kappa
  0.8349145  0.5617743

> #summary(log.model1)
> log.pred1 <- predict(log.model1, newdata = test1)
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> log.cm1 <- table(test1$Property_type, log.pred1)
> print(log.cm1)
               log.pred1
                Apartment Non-Apartment
  Apartment         24960           612
  Non-Apartment      5441          5965
> log.results1 <- confusionMatrix(log.cm1)
> log.acc1 <- log.results1$overall['Accuracy']
> print(log.acc1)
 Accuracy
0.8363081
> log.metrics1 <- binary_metrics(test1$Property_type, log.pred1)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics1)
          class   TP_rate    FP_rate precision    recall F_measure ROC_area       MCC
1     Apartment 0.9760676 0.47702963 0.8210256 0.9760676 0.8918586 0.749519 0.6027327
2 Non-Apartment 0.5229704 0.02393243 0.9069485 0.5229704 0.6634043 0.749519 0.6027327
3  weighted_avg 0.7495190 0.25048103 0.8639870 0.7495190 0.7776315 0.749519 0.6027327
```

CLASSIFIER 2: Naive Bayes

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	25442	183
Non-Apartment	9566	1794

```
Aggregating results
Selecting tuning parameters
Fitting laplace = 0, usekernel = TRUE, adjust = 1 on full training set
> n.model1
Naive Bayes

71781 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64602, 64603, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7198144  0.1477495
   TRUE      0.7432324  0.4467234

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
> #summary(n.model1)
> n.pred1 <- predict(n.model1, newdata = test1)
> n.cm1 <- table(test1$Property_type, n.pred1)
> n.results1 <- confusionMatrix(n.cm1)
> n.acc1 <- n.results1$overall['Accuracy']
> print(n.acc1)
 Accuracy
0.7925253
> n.metrics1 <- binary_metrics(test1$Property_type, n.pred1)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics1)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.8553887 0.3484131 0.8462550 0.8553887 0.8507974 0.7534878 0.5104756
2 Non-Apartment 0.6515869 0.1446113 0.6677448 0.6515869 0.6595669 0.7534878 0.5104756
3  weighted_avg 0.7534878 0.2465122 0.7569999 0.7534878 0.7551821 0.7534878 0.5104756
```

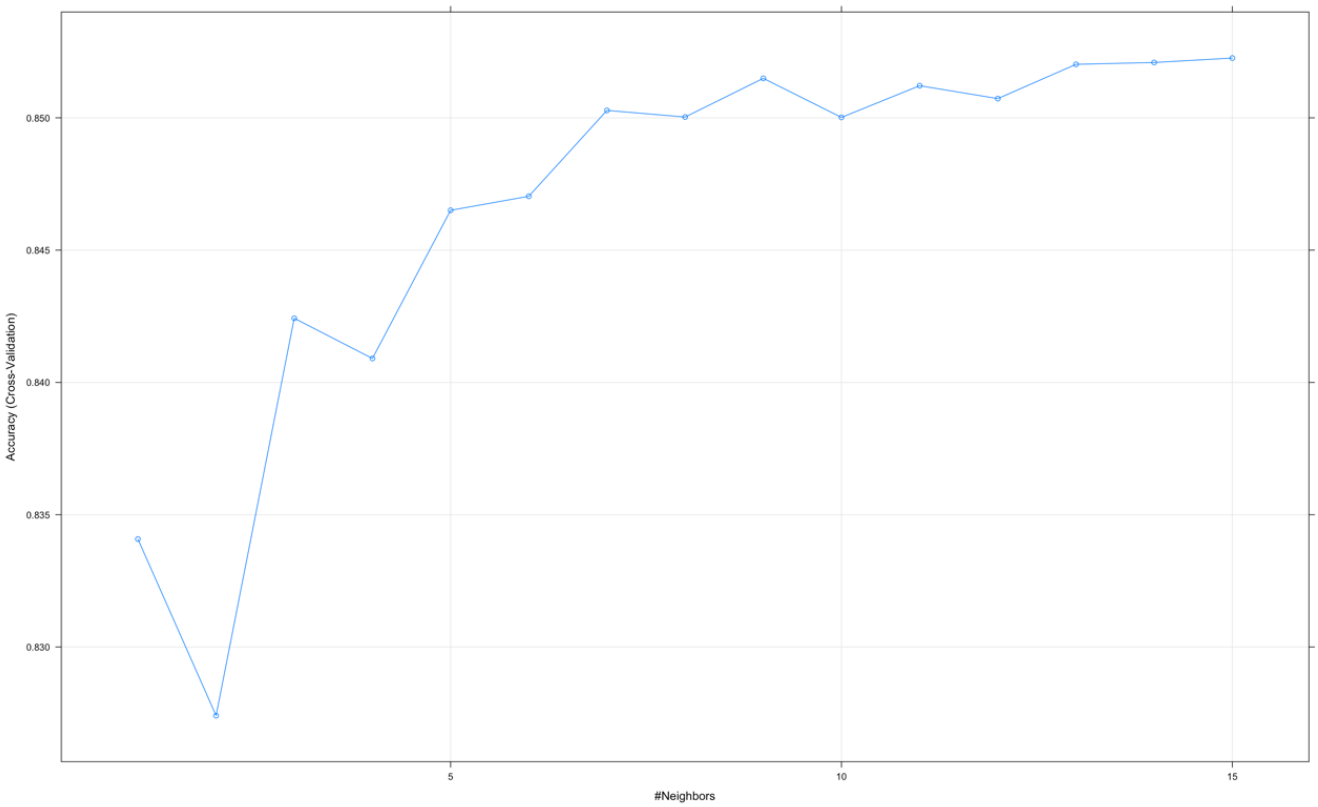

0.740
0.735
0.730
0.725
0.720
Accuracy (Cross-Validation)
FALSE
TRUE
Distribution Type

CLASSIFIER 3: K-Nearest Neighbor

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24629	884
Non-Apartment	1351	10054

```
Fitting k = 15 on full training set
> knn.model1
k-Nearest Neighbors

71781 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64602, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8340786  0.6106762
   2  0.8274056  0.5950936
   3  0.8424235  0.6228947
   4  0.8409050  0.6186321
   5  0.8465053  0.6286358
   6  0.8470347  0.6288776
   7  0.8502807  0.6347396
   8  0.8500300  0.6335175
   9  0.8514927  0.6358447
  10  0.8500160  0.6315273
  11  0.8512141  0.6336856
  12  0.8507265  0.6320622
  13  0.8520221  0.6343817
  14  0.8520918  0.6340166
  15  0.8522590  0.6341691

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
> #summary(knn.model1)
> plot(knn.model1)
> knn.pred1 <- predict(knn.model1, newdata = test1)
> knn.cm1 <- table(test1$Property_type, knn.pred1)
> knn.cm1
               knn.pred1
                Apartment Non-Apartment
  Apartment         24028          1544
  Non-Apartment      3734          7672
> knn.results1 <- confusionMatrix(knn.cm1)
> knn.acc1 <- knn.results1$overall['Accuracy']
> print(knn.acc1)
 Accuracy
0.8572665
> knn.metrics1 <- binary_metrics(test1$Property_type, knn.pred1)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics1)
          class   TP_rate    FP_rate precision    recall F_measure ROC_area      MCC
1     Apartment 0.9396215 0.32737156 0.8654996 0.9396215 0.9010387 0.806125 0.653704
2 Non-Apartment 0.6726284 0.06037854 0.8324653 0.6726284 0.7440597 0.806125 0.653704
3  weighted_avg 0.8061250 0.19387505 0.8489824 0.8061250 0.8225492 0.806125 0.653704
```


Accuracy (Cross-Validation)
0.850
0.845
0.840
0.835
0.830
5
10
15
#Neighbors

CLASSIFIER 4: Support Vector Machine

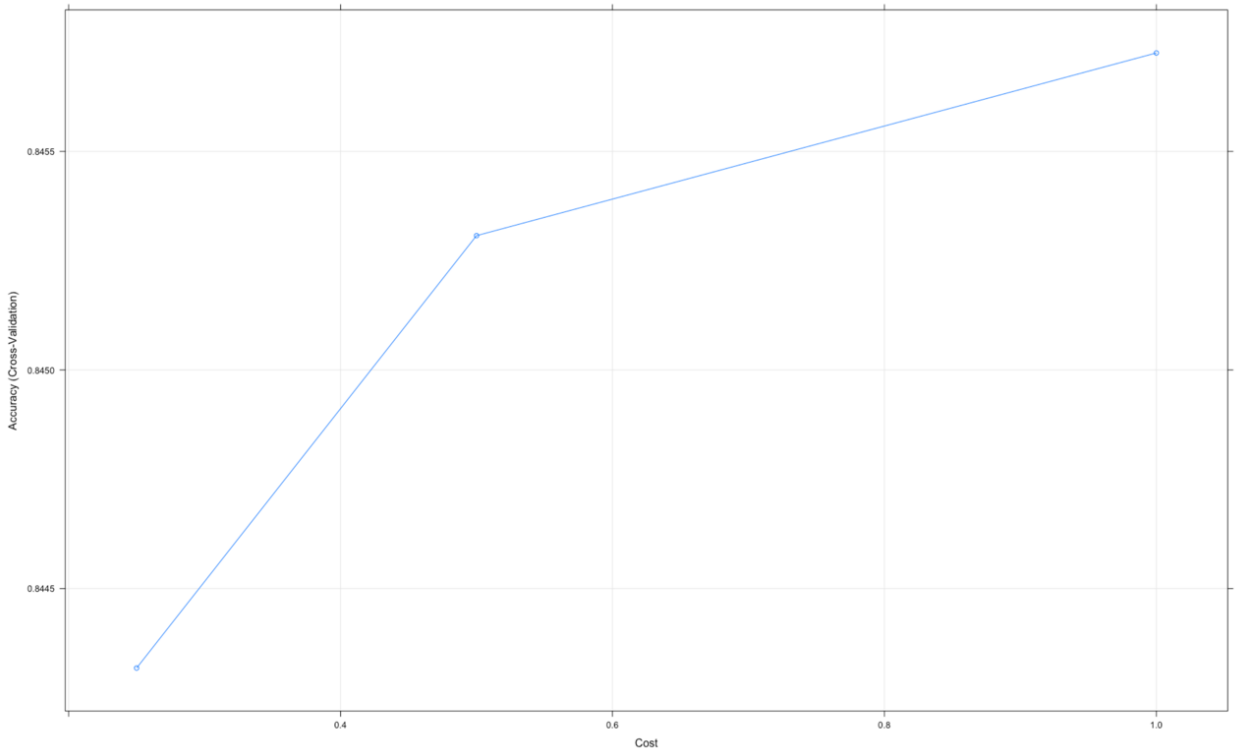
Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24910	662
Non-Apartment	4981	6425

```
+ Fold10: sigma=1520, C=1.00
- Fold10: sigma=1520, C=1.00
Aggregating results
Selecting tuning parameters
Fitting sigma = 1520, C = 1 on full training set
There were 33 warnings (use warnings() to see them)
> svm.model1
Support Vector Machines with Radial Basis Function Kernel

71781 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64602, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.8443182  0.5939050
  0.50  0.8453073  0.5956752
  1.00  0.8457252  0.5961520

Tuning parameter 'sigma' was held constant at a value of 1520.243
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 1520.243 and C = 1.
```

```
> svm.pred1 <- predict(svm.model1, newdata = test1)
> svm.cm1 <- table(test1$Property_type, svm.pred1)
> svm.cm1
              svm.pred1
               Apartment Non-Apartment
  Apartment         24910           662
  Non-Apartment      4981          6425
> # Confusion Matrix
> svm.results1 <- confusionMatrix(svm.cm1)
> svm.results1
Confusion Matrix and Statistics

              svm.pred1
               Apartment Non-Apartment
  Apartment         24910           662
  Non-Apartment      4981          6425

               Accuracy : 0.8474
                 95% CI : (0.8437, 0.851)
    No Information Rate : 0.8083
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6004

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8334
            Specificity : 0.9066
         Pos Pred Value : 0.9741
         Neg Pred Value : 0.5633
             Prevalence : 0.8083
         Detection Rate : 0.6736
   Detection Prevalence : 0.6915
      Balanced Accuracy : 0.8700

       'Positive' Class : Apartment
```

```
> # Accuracy
> svm.acc1 <- svm.results1$overall['Accuracy']
> print(svm.acc1)
 Accuracy
0.8473957
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted averages of performance
measures
> svm.metrics1 <- binary_metrics(test1$Property_type, svm.pred1)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics1)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9741123 0.43669998 0.8333612 0.9741123 0.8982565 0.7687062 0.6306018
2 Non-Apartment 0.5633000 0.02588769 0.9065895 0.5633000 0.6948575 0.7687062 0.6306018
3  weighted_avg 0.7687062 0.23129384 0.8699754 0.7687062 0.7965570 0.7687062 0.6306018
```



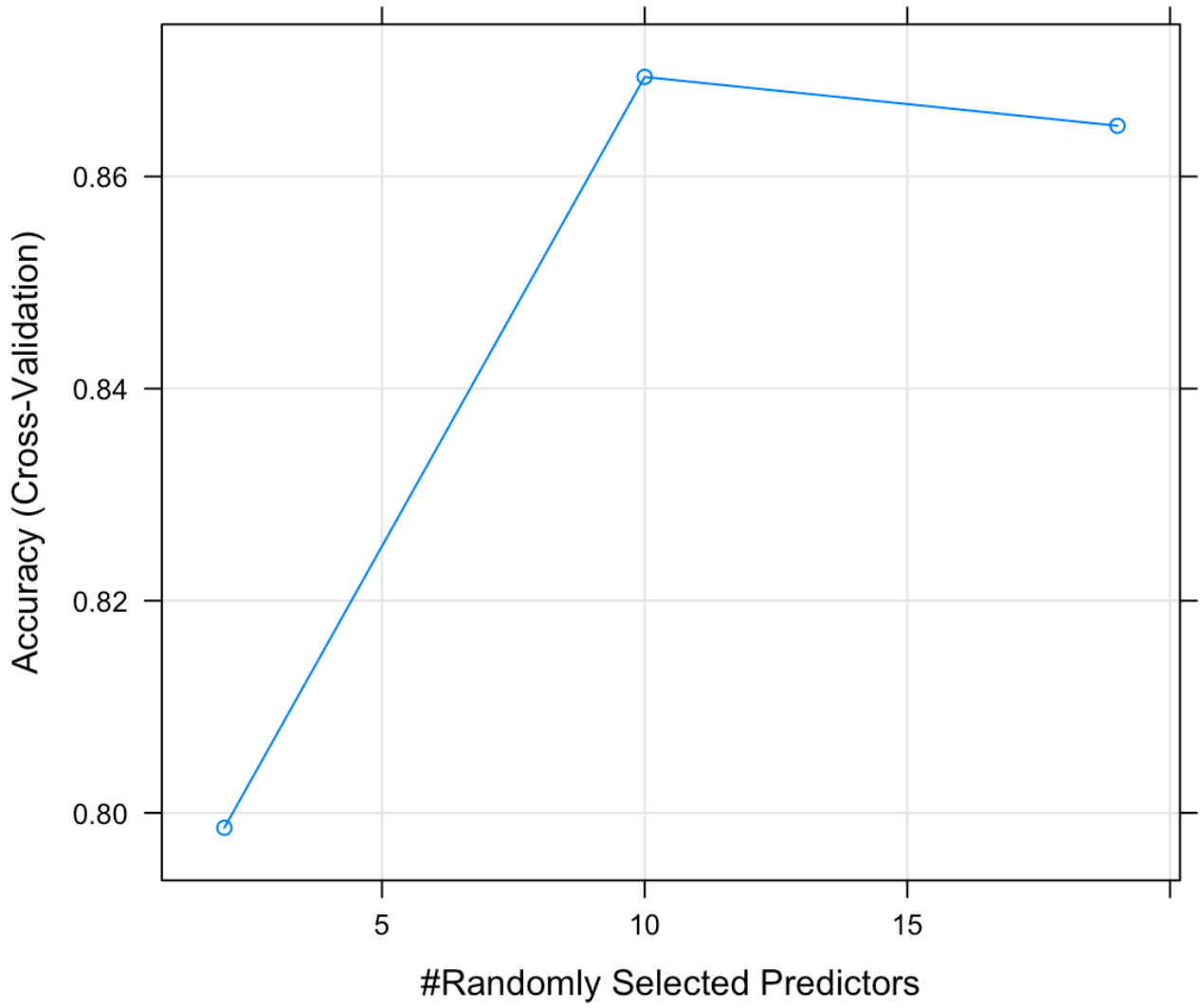
CLASSIFIER 5: Random Forest

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24647	978
Non-Apartment	3811	7549

```
> rf.model1
Random Forest

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64614, 64616, 64615, 64616, 64616, 64615, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7985932  0.4607519
  10    0.8693920  0.6703383
  19    0.8647817  0.6741493

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.
```

```
> # confusion Matrix
> rf.results1 <- confusionMatrix(rf.cm1)
> rf.results1
Confusion Matrix and Statistics

               rf.pred1
                Apartment Non-Apartment
  Apartment         24647           978
  Non-Apartment      3811          7549

               Accuracy : 0.8705
                 95% CI : (0.8671, 0.8739)
    No Information Rate : 0.7694
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6731

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8661
            Specificity : 0.8853
         Pos Pred Value : 0.9618
         Neg Pred Value : 0.6645
             Prevalence : 0.7694
         Detection Rate : 0.6664
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8757

       'Positive' Class : Apartment

> # Accuracy
> rf.acc1 <- rf.results1$overall['Accuracy']
> print(rf.acc1)
 Accuracy
0.8705151
> # Table for TP, TF..
> rf.metrics1 <- binary_metrics(test1$Property_type, rf.pred1)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics1)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9618341 0.33547535 0.8660834 0.9618341 0.9114509 0.8131794 0.6860314
2 Non-Apartment 0.6645246 0.03816585 0.8853055 0.6645246 0.7591894 0.8131794 0.6860314
3  weighted_avg 0.8131794 0.18682060 0.8756944 0.8131794 0.8353202 0.8131794 0.6860314
```


Accuracy (Cross-Validation)
0.80
0.82
0.84
0.86
5
10
15
#Randomly Selected Predictors

BORUTA FEATURE SELECTION

```
> ################################ CLASSIFIERS (BORUTA FEATURE SELECTION) ################################
> X <- train[,-1]
> y <- train[,1]
> # Train Boruta model
> boruta_model <- Boruta(X, y)
> # Print selected features
> boruta_model
Boruta performed 11 iterations in 7.709743 mins.
 12 attributes confirmed important: City_name, is_furnished, is_ready_to_move, is_RERA_registered, Latitude
and 7 more;
 No attributes deemed unimportant.
> print(boruta_model$finalDecision)
         Property_status      Price_per_unit_area Property_building_status                City_name
               Confirmed                Confirmed                Confirmed                Confirmed
               No_of_BHK                Longitude                 Latitude                    Price
               Confirmed                Confirmed                Confirmed                Confirmed
                    Size             is_furnished       is_RERA_registered         is_ready_to_move
               Confirmed                Confirmed                Confirmed                Confirmed
Levels: Tentative Confirmed Rejected
> # Print the selected features
> selected_features <- (which(boruta_model$finalDecision == "Confirmed"))
> print(selected_features)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	89.49%
Naive Bayes	85.69%
k-NN	93.92%
Support Vector Machine	90.62%
Random Forest	95.51%

CLASSIFIER 1: GLM Model (Logistic regression)

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24246	1326
Non-Apartment	2557	8849

```
- Fold09: parameter=none
+ Fold10: parameter=none
- Fold10: parameter=none
Aggregating results
Fitting final model on full training set
There were 21 warnings (use warnings() to see them)
> log.model2
Generalized Linear Model

71781 samples
   11 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64602, 64603, 64603, 64603, ...
Resampling results:

  Accuracy   Kappa
  0.8991377  0.7571553

> plot(log.model2)
Error in plot.train(log.model2) :
  There are no tuning parameters for this model.
> #summary(log.model2)
> log.pred2 <- predict(log.model2, newdata = test2)
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> log.cm2 <- table(test2$Property_type, log.pred2)
> print(log.cm2)
               log.pred2
                Apartment Non-Apartment
  Apartment         24246          1326
  Non-Apartment      2557          8849
```

```
> # Confusion Matrix
> log.results2 <- confusionMatrix(log.cm2)
> log.results2
Confusion Matrix and Statistics

                log.pred2
                 Apartment Non-Apartment
  Apartment          24246          1326
  Non-Apartment       2557          8849

               Accuracy : 0.895
                 95% CI : (0.8918, 0.8981)
    No Information Rate : 0.7248
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7463

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9046
            Specificity : 0.8697
         Pos Pred Value : 0.9481
         Neg Pred Value : 0.7758
             Prevalence : 0.7248
         Detection Rate : 0.6557
   Detection Prevalence : 0.6915
      Balanced Accuracy : 0.8871

       'Positive' Class : Apartment
```

```
> # Accuracy
> log.acc2 <- log.results2$overall['Accuracy']
> print(log.acc2)
 Accuracy
0.8949916
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cla
measures
> log.metrics2 <- binary_metrics(test2$Property_type, log.pred2)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics2)
         class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1    Apartment 0.9481464 0.22418026 0.9046002 0.9481464 0.9258616 0.8619831 0.7487009
2 Non-Apartment 0.7758197 0.05185359 0.8696806 0.7758197 0.8200732 0.8619831 0.7487009
3  weighted_avg 0.8619831 0.13801692 0.8871404 0.8619831 0.8729674 0.8619831 0.7487009
```

CLASSIFIER 2: Naive Bayes

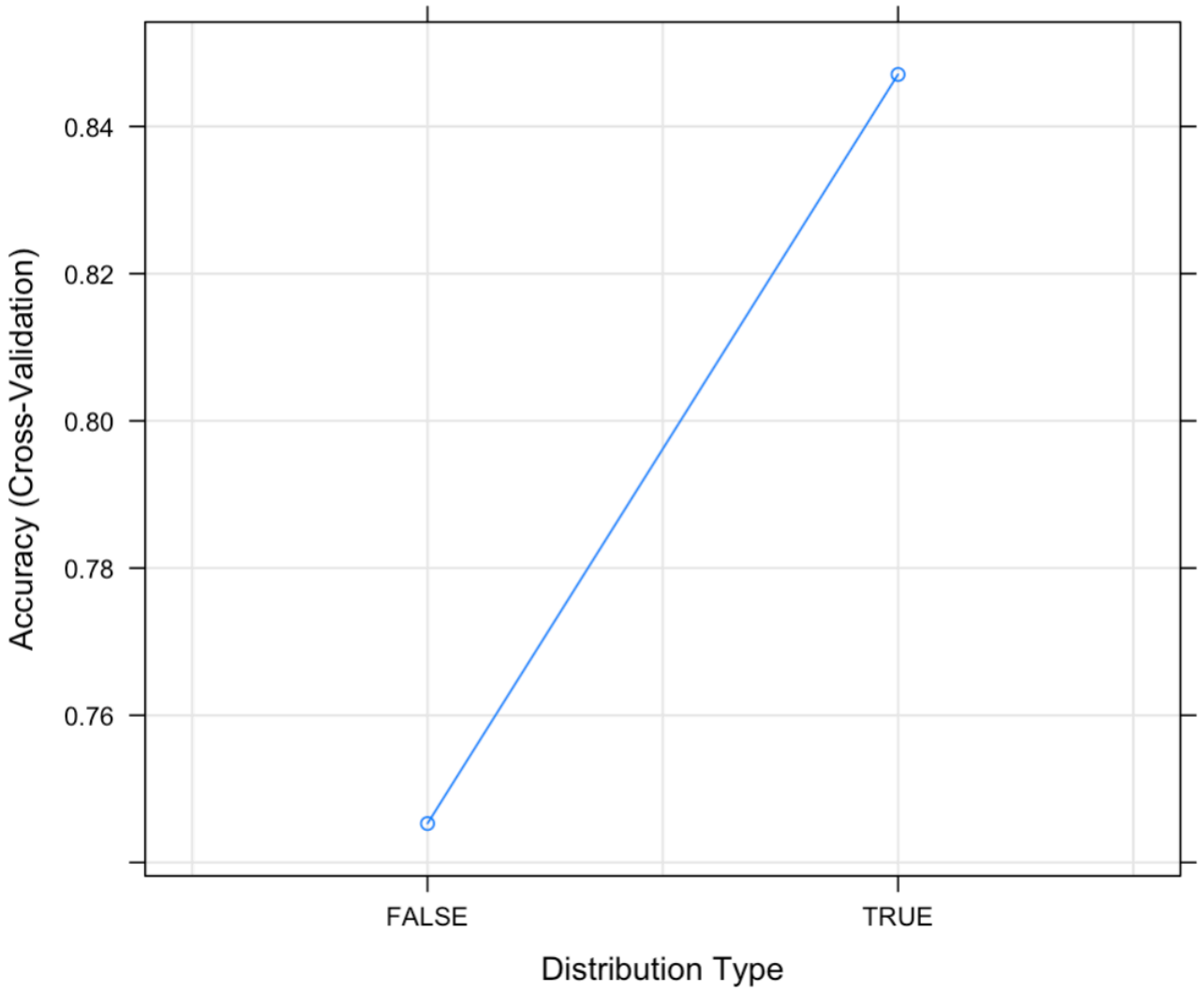
Confusion Matrix:

	Apartment	Non-Apartment
Apartment	22227	3345
Non-Apartment	1945	9461

```
+ Fold10: usekernel=FALSE, laplace=0, adjust=1
- Fold10: usekernel=FALSE, laplace=0, adjust=1
Aggregating results
Selecting tuning parameters
Fitting laplace = 0, usekernel = TRUE, adjust = 1 on full training set
> n.model2
Naive Bayes

71781 samples
   11 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64602, 64603, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7452947  0.2584988
   TRUE      0.8470625  0.6585783

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
```

```
> # Confusion Matrix
> n.results2 <- confusionMatrix(n.cm2)
> n.results2
Confusion Matrix and Statistics

              n.pred2
               Apartment Non-Apartment
  Apartment        22227          3345
  Non-Apartment     1945          9461

               Accuracy : 0.8569
                 95% CI : (0.8533, 0.8605)
    No Information Rate : 0.6537
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6757

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9195
            Specificity : 0.7388
         Pos Pred Value : 0.8692
         Neg Pred Value : 0.8295
             Prevalence : 0.6537
         Detection Rate : 0.6011
   Detection Prevalence : 0.6915
      Balanced Accuracy : 0.8292

       'Positive' Class : Apartment
```

```
> # Accuracy
> n.acc2 <- n.results2$overall['Accuracy']
> print(n.acc2)
Accuracy
0.856942
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted averages of performance
measures
> n.metrics2 <- binary_metrics(test2$Property_type, n.pred2)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics2)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.8691929 0.1705243 0.9195350 0.8691929 0.8936555 0.8493343 0.6781991
2 Non-Apartment 0.8294757 0.1308071 0.7387943 0.8294757 0.7815133 0.8493343 0.6781991
3  weighted_avg 0.8493343 0.1506657 0.8291647 0.8493343 0.8375844 0.8493343 0.6781991
```


Accuracy (Cross-Validation)
0.84
0.82
0.80
0.78
0.76
FALSE
TRUE
Distribution Type

CLASSIFIER 3: K-Nearest Neighbor

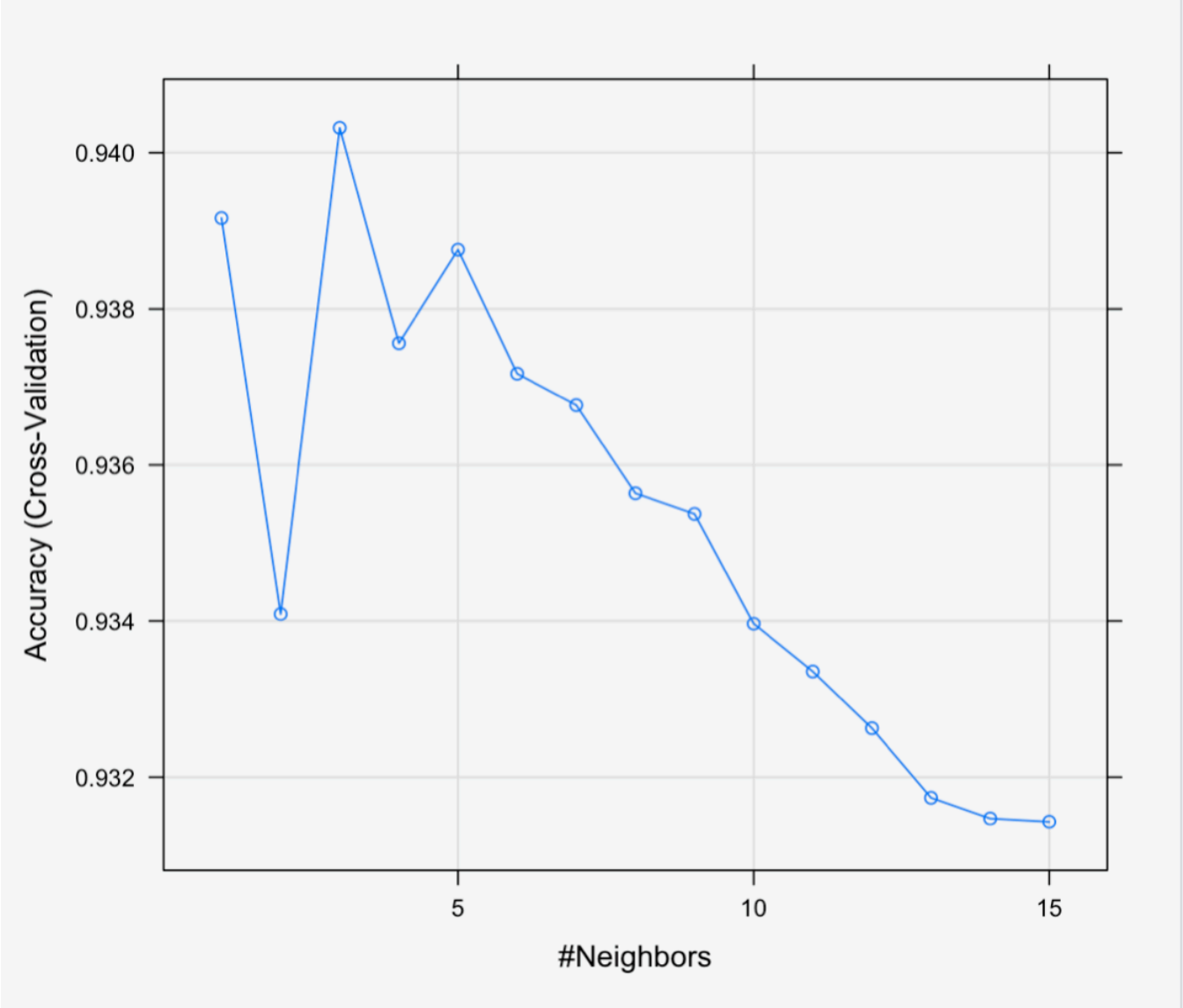
Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24666	906
Non-Apartment	1339	10067

```
> knn.model2
k-Nearest Neighbors

71781 samples
   11 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64603, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9391621  0.8565354
   2  0.9340912  0.8445020
   3  0.9403184  0.8583762
   4  0.9375600  0.8518400
   5  0.9387581  0.8544564
   6  0.9371700  0.8506024
   7  0.9367659  0.8495812
   8  0.9356375  0.8467837
   9  0.9353728  0.8460932
  10  0.9339658  0.8426330
  11  0.9333528  0.8411019
  12  0.9326284  0.8392787
  13  0.9317368  0.8370817
  14  0.9314721  0.8364410
  15  0.9314303  0.8363359

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 3.
```

```
> # Confusion Matrix
> knn.results2 <- confusionMatrix(knn.cm2)
> knn.results2
Confusion Matrix and Statistics

                knn.pred2
                 Apartment Non-Apartment
  Apartment          24666           906
  Non-Apartment       1339         10067

               Accuracy : 0.9393
                 95% CI : (0.9368, 0.9417)
    No Information Rate : 0.7033
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8562

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9485
            Specificity : 0.9174
         Pos Pred Value : 0.9646
         Neg Pred Value : 0.8826
             Prevalence : 0.7033
         Detection Rate : 0.6670
   Detection Prevalence : 0.6915
      Balanced Accuracy : 0.9330

       'Positive' Class : Apartment
```

```
> # Accuracy
> knn.acc2 <- knn.results2$overall['Accuracy']
> print(knn.acc2)
 Accuracy
0.9392882
> # Accuracy
> knn.acc2 <- knn.results2$overall['Accuracy']
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted averages of performance
measures
> knn.metrics2 <- binary_metrics(test2$Property_type, knn.pred2)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics2)
         class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9645706 0.11739435 0.9485099 0.9645706 0.9564728 0.9235881 0.8565085
2 Non-Apartment 0.8826056 0.03542938 0.9174337 0.8826056 0.8996827 0.9235881 0.8565085
3  weighted_avg 0.9235881 0.07641186 0.9329718 0.9235881 0.9280778 0.9235881 0.8565085
>
```



CLASSIFIER 4: Support Vector Machine

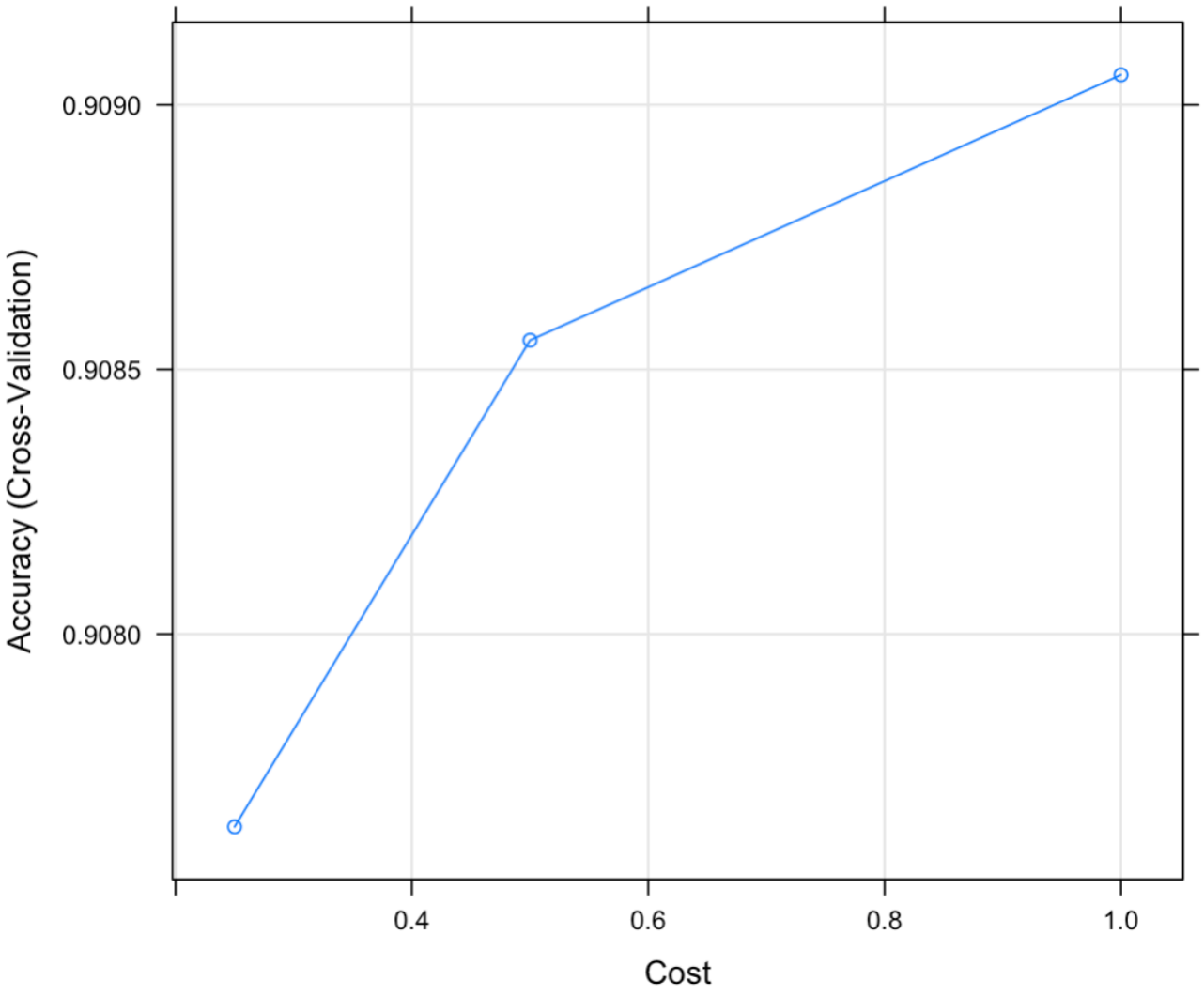
Confusion Matrix:

```
+ Fold10: sigma=0.127, C=0.50
- Fold10: sigma=0.127, C=0.50
+ Fold10: sigma=0.127, C=1.00
- Fold10: sigma=0.127, C=1.00
Aggregating results
Selecting tuning parameters
Fitting sigma = 0.127, C = 1 on full training set
There were 32 warnings (use warnings() to see them)
> svm.model2
Support Vector Machines with Radial Basis Function Kernel

71781 samples
   11 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64603, 64603, 64603, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.9076357  0.7789153
  0.50  0.9085552  0.7810704
  1.00  0.9090567  0.7822361

Tuning parameter 'sigma' was held constant at a value of 0.1269765
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1269765 and C = 1.
```

	Apartment	Non-Apartment
Apartment	24272	1300
Non-Apartment	2167	9239

```
> # Accuracy
> svm.acc2 <- svm.results2$overall['Accuracy']
> print(svm.acc2)
 Accuracy
0.9062415
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted averages of performance
measures
> svm.metrics2 <- binary_metrics(test2$Property_type, svm.pred2)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics2)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area      MCC
1     Apartment 0.9491631 0.18998773 0.9180377 0.9491631 0.9333410 0.8795877 0.776728
2 Non-Apartment 0.8100123 0.05083685 0.8766486 0.8100123 0.8420141 0.8795877 0.776728
3  weighted_avg 0.8795877 0.12041229 0.8973432 0.8795877 0.8876776 0.8795877 0.776728
```

```
> # Confusion Matrix
```

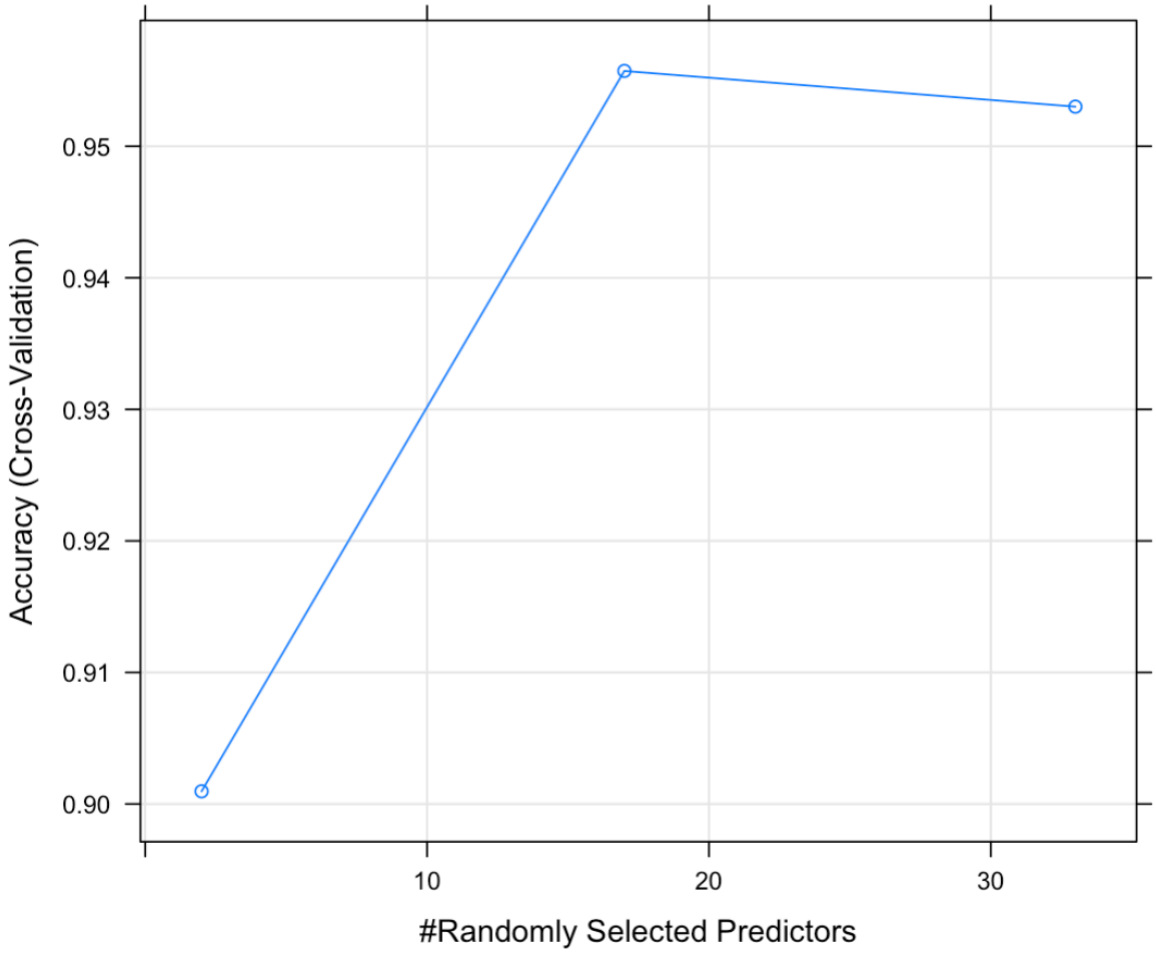


CLASSIFIER 5: Random Forest

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24861	711
Non-Apartment	947	10459

```
> rf.model2
Random Forest

71781 samples
   11 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64602, 64603, 64603, 64603, 64603, 64603, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9009626  0.7573352
  17    0.9557265  0.8956590
  33    0.9530099  0.8892354

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 17.
```


Accuracy (Cross-Validation)
0.90
0.91
0.92
0.93
0.94
0.95
10
20
30
#Randomly Selected Predictors

```
> # Confusion Matrix
> rf.results2 <- confusionMatrix(rf.cm2)
> rf.results2
Confusion Matrix and Statistics

               rf.pred2
                Apartment Non-Apartment
  Apartment         24861           711
  Non-Apartment       947         10459

               Accuracy : 0.9552
                 95% CI : (0.953, 0.9572)
    No Information Rate : 0.6979
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8943

 Mcnemar's Test P-Value : 7.865e-09

            Sensitivity : 0.9633
            Specificity : 0.9363
         Pos Pred Value : 0.9722
         Neg Pred Value : 0.9170
             Prevalence : 0.6979
         Detection Rate : 0.6723
   Detection Prevalence : 0.6915
      Balanced Accuracy : 0.9498

       'Positive' Class : Apartment

> # Accuracy
> rf.acc2 <- rf.results2$overall['Accuracy']
> print(rf.acc2)
 Accuracy
0.9551625
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cla
measures
> rf.metrics2 <- binary_metrics(test2$Property_type, rf.pred2)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics2)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9721962 0.08302648 0.9633060 0.9721962 0.9677306 0.9445848 0.8943961
2 Non-Apartment 0.9169735 0.02780385 0.9363474 0.9169735 0.9265592 0.9445848 0.8943961
3  weighted_avg 0.9445848 0.05541516 0.9498267 0.9445848 0.9471449 0.9445848 0.8943961
```

CHI-SQUARE FEATURE SELECTION

```
> # The features with higher chi-square values are considered more important.
> sort(chi_values, decreasing=TRUE)[1:5]
          Latitude.X-squared          Longitude.X-squared Price_per_unit_area.X-squared
                    60431.72                     59742.16                     30534.08
             Price.X-squared                Size.X-squared
                    26703.82                     24452.04
```

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	83.49%
Naive Bayes	73.64%
k-NN	85.52%
Support Vector Machine	84.87%
Random Forest	87.11%

CLASSIFIER 1: GLM Model (Logistic regression)

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24973	652
Non-Apartment	5452	5908

```
> log.model3
Generalized Linear Model

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64615, 64616, 64615, 64616, 64615, 64615, ...
Resampling results:

  Accuracy   Kappa
  0.8349327  0.5594948
```

```
> # Confusion Matrix
> log.results3 <- confusionMatrix(log.cm3)
> log.results3
Confusion Matrix and Statistics

              log.pred3
               Apartment Non-Apartment
  Apartment         24973           652
  Non-Apartment      5452          5908

               Accuracy : 0.835
                 95% CI : (0.8311, 0.8387)
    No Information Rate : 0.8226
    P-Value [Acc > NIR] : 1.959e-10

                  Kappa : 0.5606

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8208
            Specificity : 0.9006
         Pos Pred Value : 0.9746
         Neg Pred Value : 0.5201
             Prevalence : 0.8226
         Detection Rate : 0.6752
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8607

       'Positive' Class : Apartment

> # Accuracy
> log.acc3 <- log.results3$overall['Accuracy']
> print(log.acc3)
 Accuracy
0.8349601
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each
erformance measures
> log.metrics3 <- binary_metrics(test3$Property_type, log.pred3)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics3)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9745561 0.4799296 0.8208053 0.9745561 0.8910972 0.7473133 0.5973533
2 Non-Apartment 0.5200704 0.0254439 0.9006098 0.5200704 0.6593750 0.7473133 0.5973533
3  weighted_avg 0.7473133 0.2526867 0.8607075 0.7473133 0.7752361 0.7473133 0.5973533
```

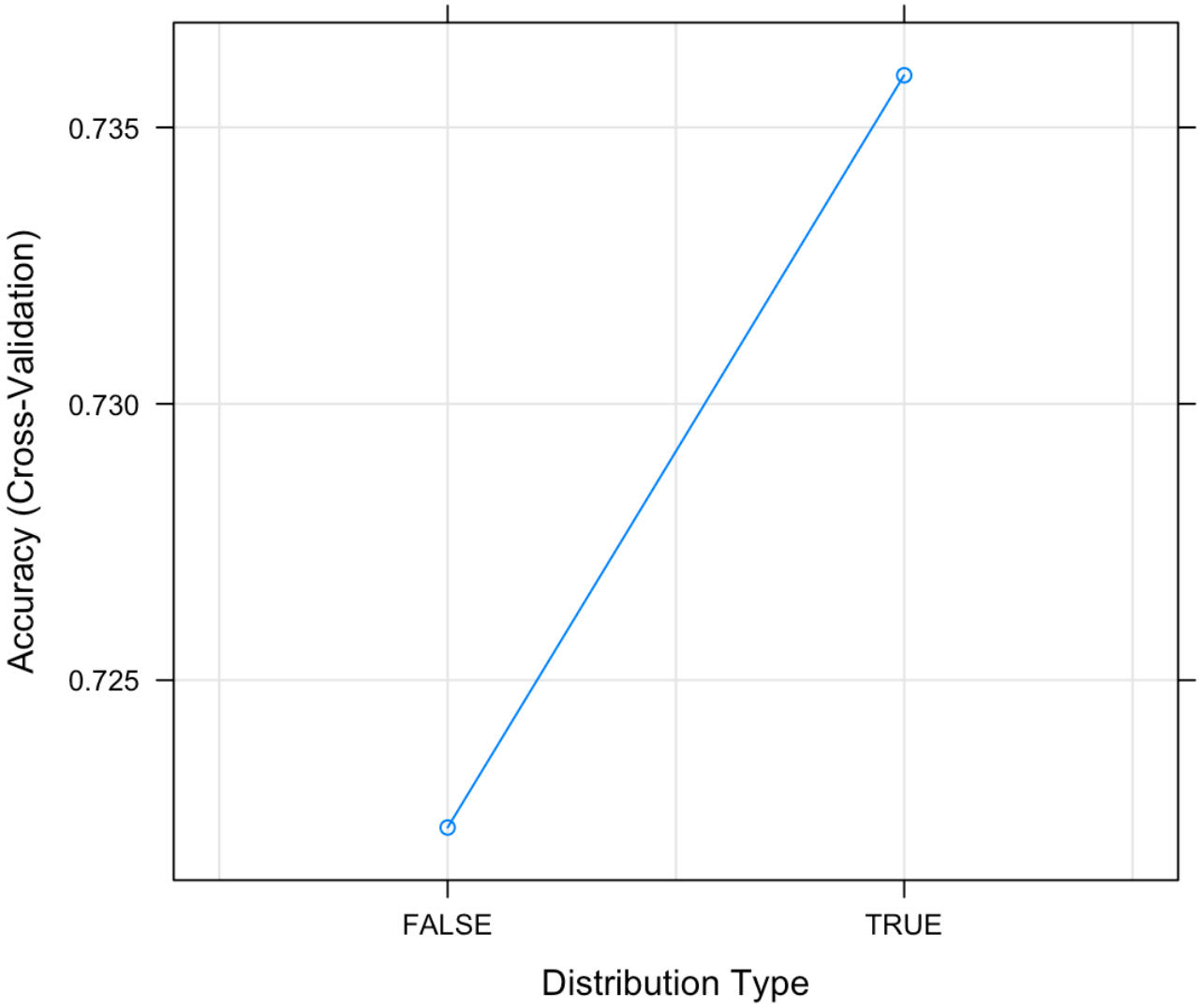
CLASSIFIER 2: Naive Bayes

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	25442	183
Non-Apartment	9566	1794

```
> n.model3
Naive Bayes

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64616, 64614, 64616, 64616, 64615, 64615, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7223343  0.1600923
   TRUE      0.7359431  0.1957573

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
```

```
> n.results3
Confusion Matrix and Statistics

               n.pred3
                Apartment Non-Apartment
  Apartment         25442           183
  Non-Apartment      9566          1794

               Accuracy : 0.7364
                 95% CI : (0.7319, 0.7409)
    No Information Rate : 0.9465
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1958

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7267
            Specificity : 0.9074
         Pos Pred Value : 0.9929
         Neg Pred Value : 0.1579
             Prevalence : 0.9465
         Detection Rate : 0.6879
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8171

       'Positive' Class : Apartment

> # Accuracy
> n.acc3 <- n.results3$overall['Accuracy']
> print(n.acc3)
 Accuracy
0.7364067
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cla
erformance measures
> n.metrics3 <- binary_metrics(test3$Property_type, n.pred3)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics3)
          class   TP_rate     FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9928585 0.842077465 0.7267482 0.9928585 0.8392130 0.5753905 0.3092295
2 Non-Apartment 0.1579225 0.007141463 0.9074355 0.1579225 0.2690260 0.5753905 0.3092295
3  weighted_avg 0.5753905 0.424609464 0.8170918 0.5753905 0.5541195 0.5753905 0.3092295
```


0.735
0.730
0.725
Accuracy (Cross-Validation)
FALSE
TRUE
Distribution Type

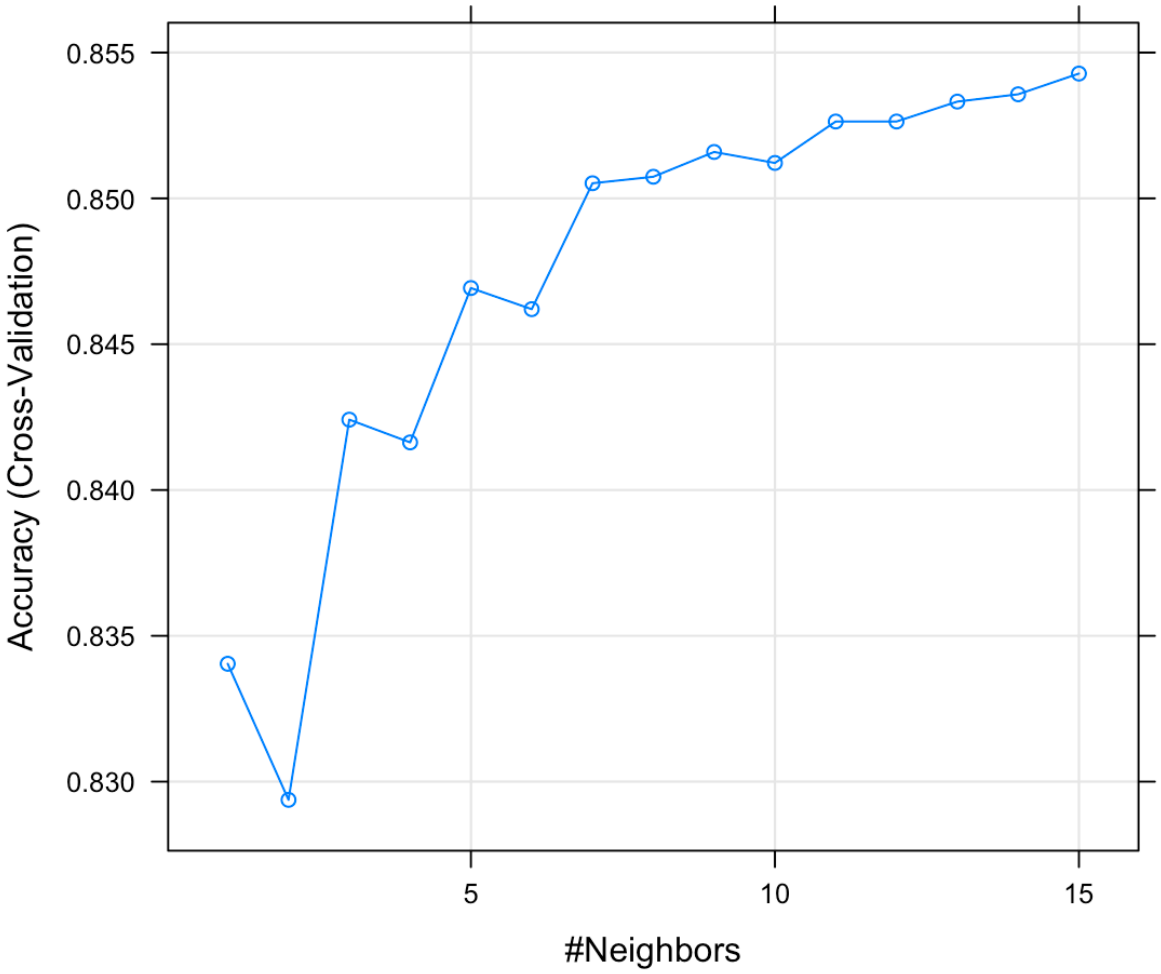
CLASSIFIER 3: K-Nearest Neighbor

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24039	1586
Non-Apartment	3767	7593

```
> knn.model3
k-Nearest Neighbors

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64615, 64615, 64616, 64616, 64615, 64615, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8340415  0.6099113
   2  0.8293754  0.5985450
   3  0.8424126  0.6220165
   4  0.8416325  0.6196307
   5  0.8469254  0.6281432
   6  0.8462010  0.6254932
   7  0.8505189  0.6341804
   8  0.8507418  0.6340682
   9  0.8515914  0.6345769
  10  0.8512154  0.6329963
  11  0.8526361  0.6353345
  12  0.8526361  0.6353427
  13  0.8533186  0.6359835
  14  0.8535693  0.6365108
  15  0.8542796  0.6377742

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
```

```
> knn.results3
Confusion Matrix and Statistics

              knn.pred3
               Apartment Non-Apartment
  Apartment         24039          1586
  Non-Apartment      3767          7593

               Accuracy : 0.8553
                 95% CI : (0.8516, 0.8588)
    No Information Rate : 0.7518
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6407

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8645
            Specificity : 0.8272
         Pos Pred Value : 0.9381
         Neg Pred Value : 0.6684
             Prevalence : 0.7518
         Detection Rate : 0.6500
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8459

       'Positive' Class : Apartment

> # Accuracy
> knn.acc3 <- knn.results3$overall['Accuracy']
> print(knn.acc3)
 Accuracy
0.8552656
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and weighted a
erformance measures
> knn.metrics3 <- binary_metrics(test3$Property_type, knn.pred3)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics3)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9381073 0.33160211 0.8645256 0.9381073 0.8998147 0.8032526 0.6477221
2 Non-Apartment 0.6683979 0.06189268 0.8272143 0.6683979 0.7393739 0.8032526 0.6477221
3  weighted_avg 0.8032526 0.19674740 0.8458700 0.8032526 0.8195943 0.8032526 0.6477221
```


0.855
0.850
0.845
0.840
0.835
0.830
Accuracy (Cross-Validation)
5
10
15
#Neighbors

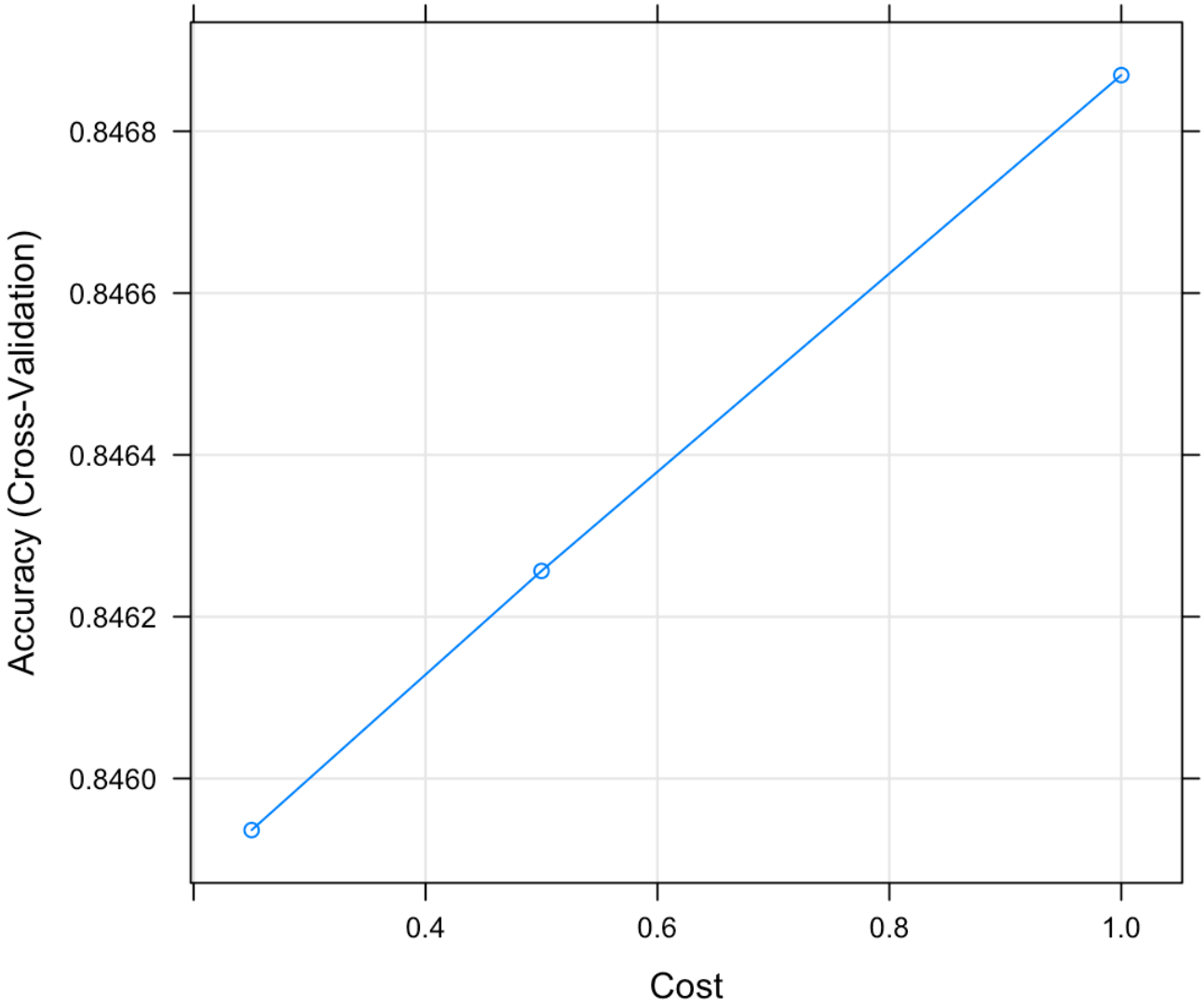
CLASSIFIER 4: Support Vector Machine

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24805	820
Non-Apartment	4775	6585

```
> svm.model3
Support Vector Machines with Radial Basis Function Kernel

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64616, 64615, 64615, 64615, 64616, 64616, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.8459362  0.5989171
  0.50  0.8462566  0.5992851
  1.00  0.8468694  0.6007035

Tuning parameter 'sigma' was held constant at a value of 1545.108
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 1545.108 and C = 1.
```

```
> svm.results3
Confusion Matrix and Statistics

               svm.pred3
                Apartment Non-Apartment
  Apartment         24805           820
  Non-Apartment      4775          6585

               Accuracy : 0.8487
                 95% CI : (0.845, 0.8524)
    No Information Rate : 0.7998
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6064

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8386
            Specificity : 0.8893
         Pos Pred Value : 0.9680
         Neg Pred Value : 0.5797
             Prevalence : 0.7998
         Detection Rate : 0.6707
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8639

       'Positive' Class : Apartment

> # Accuracy
> svm.acc3 <- svm.results3$overall['Accuracy']
> print(svm.acc3)
 Accuracy
0.8487225
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each c
erformance measures
> svm.metrics3 <- binary_metrics(test3$Property_type, svm.pred3)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics3)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9680000 0.4203345 0.8385734 0.9680000 0.8986505 0.7738327 0.6313568
2 Non-Apartment 0.5796655 0.0320000 0.8892640 0.5796655 0.7018385 0.7738327 0.6313568
3  weighted_avg 0.7738327 0.2261673 0.8639187 0.7738327 0.8002445 0.7738327 0.6313568
```


Accuracy (Cross-Validation)
0.8468
0.8466
0.8464
0.8462
0.8460
0.4
0.6
0.8
1.0
Cost

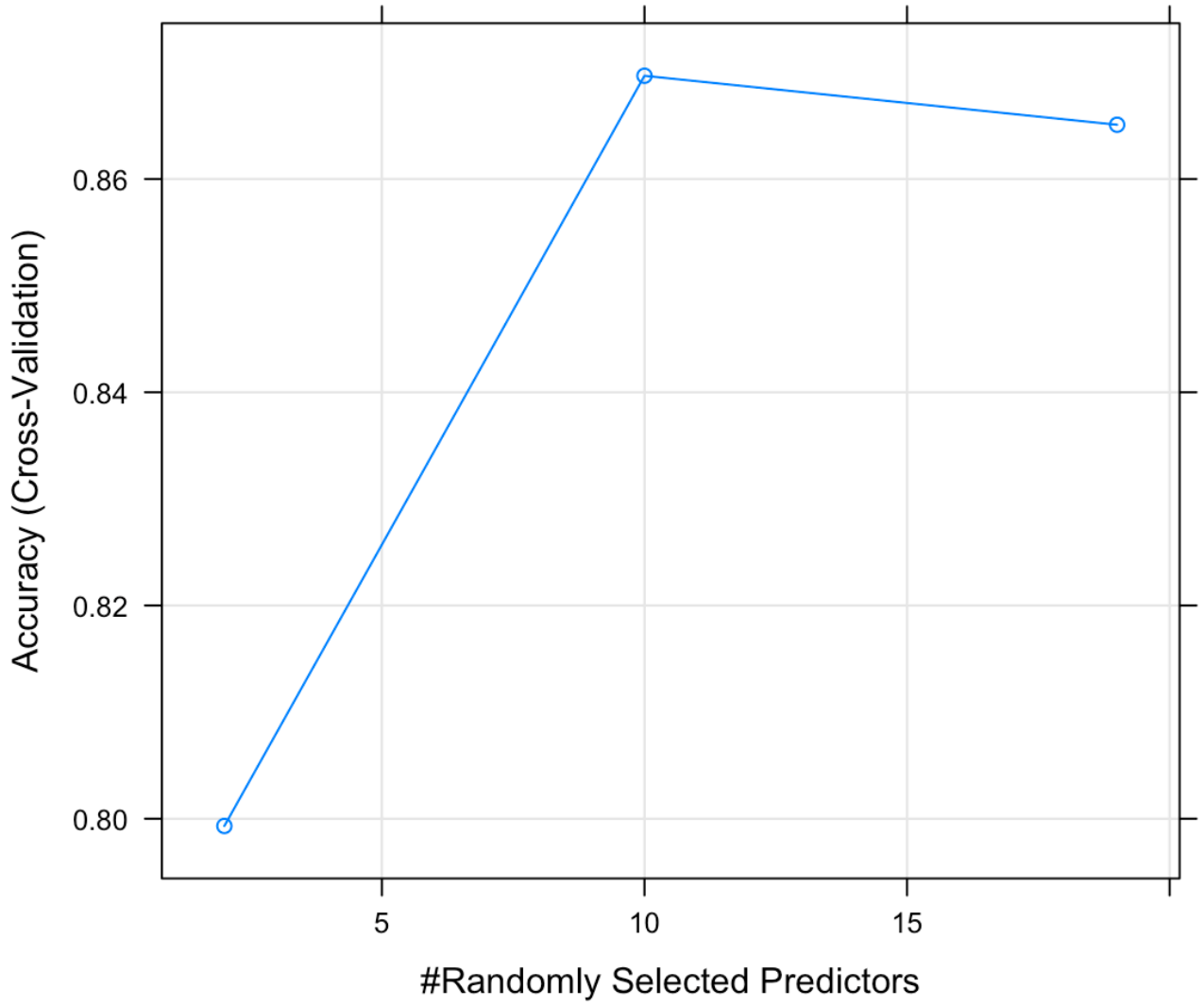
CLASSIFIER 5: Random Forest

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24659	966
Non-Apartment	3800	7560

```
> rf.model3
Random Forest

71795 samples
    4 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64616, 64616, 64616, 64615, 64615, 64616, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7993175  0.4622232
  10    0.8696845  0.6711335
  19    0.8650880  0.6749791

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.
```

```
> rf.results3
Confusion Matrix and Statistics

               rf.pred3
                Apartment Non-Apartment
  Apartment         24659           966
  Non-Apartment      3800          7560

               Accuracy : 0.8711
                 95% CI : (0.8677, 0.8745)
    No Information Rate : 0.7695
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6746

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8665
            Specificity : 0.8867
         Pos Pred Value : 0.9623
         Neg Pred Value : 0.6655
             Prevalence : 0.7695
         Detection Rate : 0.6667
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.8766

       'Positive' Class : Apartment

> # Accuracy
> rf.acc3 <- rf.results3$overall['Accuracy']
> print(rf.acc3)
 Accuracy
0.8711369
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each clas
erformance measures
> rf.metrics3 <- binary_metrics(test3$Property_type, rf.pred3)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics3)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9623024 0.33450704 0.8664746 0.9623024 0.9118778 0.8138977 0.6876331
2 Non-Apartment 0.6654930 0.03769756 0.8866995 0.6654930 0.7603339 0.8138977 0.6876331
3  weighted_avg 0.8138977 0.18610230 0.8765870 0.8138977 0.8361059 0.8138977 0.6876331
```


Accuracy (Cross-Validation)
0.80
0.82
0.84
0.86
5
10
15
#Randomly Selected Predictors

R-PART IMPORTANCE FEATURE SELECTION

```
> rPartMod <- train(Property_type ~ ., data=train, method="rpart")
> rpartImp <- varImp(rPartMod)
> print(rpartImp)
rpart variable importance

  only 20 most important variables shown (out of 36)

                                  Overall
Longitude                         100.000
City_nameMumbai                    87.822
Property_building_statusUNVERIFIED 65.419
Latitude                           65.297
Price_per_unit_area                33.986
No_of_BHK2 BHK                     32.664
Size                               32.397
Price                               8.137
is_RERA_registeredTRUE              0.000
`No_of_BHK6 BHK`                    0.000
`No_of_BHK14 BHK`                   0.000
`No_of_BHK9 BHK`                    0.000
`No_of_BHK3 BHK`                    0.000
City_nameChennai                    0.000
`No_of_BHK4 BHK`                    0.000
`No_of_BHK7 BHK`                    0.000
`No_of_BHK11 BHK`                   0.000
is_furnishedUnfurnished             0.000
City_nameKolkata                    0.000
City_nameHyderabad                  0.000
```

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	87.66%
Naive Bayes	86.00%
k-NN	94.27%
Support Vector Machine	90.46%
Random Forest	95.21%

CLASSIFIER 1: GLM Model (Logistic regression)

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24284	1266
Non-Apartment	3286	8071

```
> log.model4
Generalized Linear Model

71643 samples
    7 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64479, 64479, 64478, 64478, 64479, 64479, ...
Resampling results:

  Accuracy   Kappa
  0.8743492  0.6896358
```

```
> log.pred4 <- predict(log.model4, newdata = test4)
> log.cm4 <- table(test4$Property_type, log.pred4)
> # Confusion Matrix
> log.results4 <- confusionMatrix(log.cm4)
> log.results4
Confusion Matrix and Statistics

               log.pred4
                Apartment Non-Apartment
  Apartment         24284          1266
  Non-Apartment      3286          8071

               Accuracy : 0.8767
                 95% CI : (0.8733, 0.88)
    No Information Rate : 0.747
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6955

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8808
            Specificity : 0.8644
         Pos Pred Value : 0.9505
         Neg Pred Value : 0.7107
             Prevalence : 0.7470
         Detection Rate : 0.6580
   Detection Prevalence : 0.6923
      Balanced Accuracy : 0.8726

       'Positive' Class : Apartment

> # Accuracy
> log.acc4 <- log.results4$overall['Accuracy']
> print(log.acc4)
Accuracy
0.876663
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class and
 measures
> log.metrics4 <- binary_metrics(test4$Property_type, log.pred4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics4)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9504501 0.2893370 0.8808125 0.9504501 0.9143072 0.8305566 0.7019093
2 Non-Apartment 0.7106630 0.0495499 0.8644104 0.7106630 0.7800329 0.8305566 0.7019093
3  weighted_avg 0.8305566 0.1694434 0.8726114 0.8305566 0.8471700 0.8305566 0.7019093
```

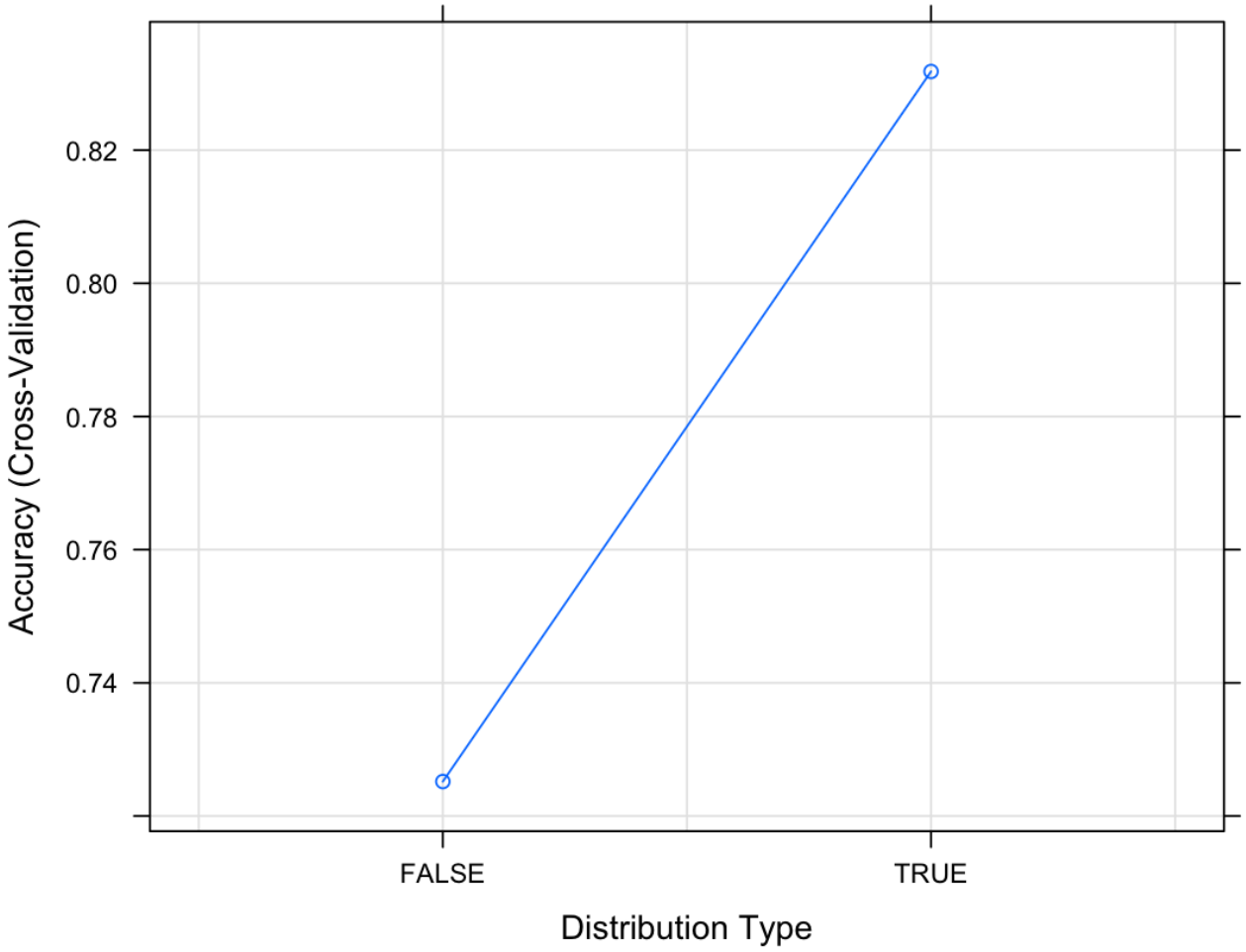
CLASSIFIER 2: Naive Bayes

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	21780	3770
Non-Apartment	1395	9962

```
> n.model4
Naive Bayes

71643 samples
    7 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64478, 64479, 64480, 64478, 64479, 64478, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7251789  0.1658130
   TRUE      0.8318064  0.6191198

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
```

```
> plot(n.model4)
> n.pred4 <- predict(n.model4, newdata = test4)
> n.cm4 <- table(test4$Property_type, n.pred4)
> # Confusion Matrix
> n.results4 <- confusionMatrix(n.cm4)
> n.results4
Confusion Matrix and Statistics

               n.pred4
                Apartment Non-Apartment
  Apartment         21780          3770
  Non-Apartment      1395          9962

               Accuracy : 0.8601
                 95% CI : (0.8565, 0.8636)
    No Information Rate : 0.6279
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6896

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9398
            Specificity : 0.7255
         Pos Pred Value : 0.8524
         Neg Pred Value : 0.8772
             Prevalence : 0.6279
         Detection Rate : 0.5901
   Detection Prevalence : 0.6923
      Balanced Accuracy : 0.8326

       'Positive' Class : Apartment

> # Accuracy
> n.acc4 <- n.results4$overall['Accuracy']
> print(n.acc4)
 Accuracy
0.8600536
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class
 measures
> n.metrics4 <- binary_metrics(test4$Property_type, n.pred4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics4)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area      MCC
1     Apartment 0.8524462 0.1228317 0.9398058 0.8524462 0.8939969 0.8648072 0.696697
2 Non-Apartment 0.8771683 0.1475538 0.7254588 0.8771683 0.7941329 0.8648072 0.696697
3  weighted_avg 0.8648072 0.1351928 0.8326323 0.8648072 0.8440649 0.8648072 0.696697
```


Accuracy (Cross-Validation)
0.82
0.80
0.78
0.76
0.74
FALSE
TRUE
Distribution Type

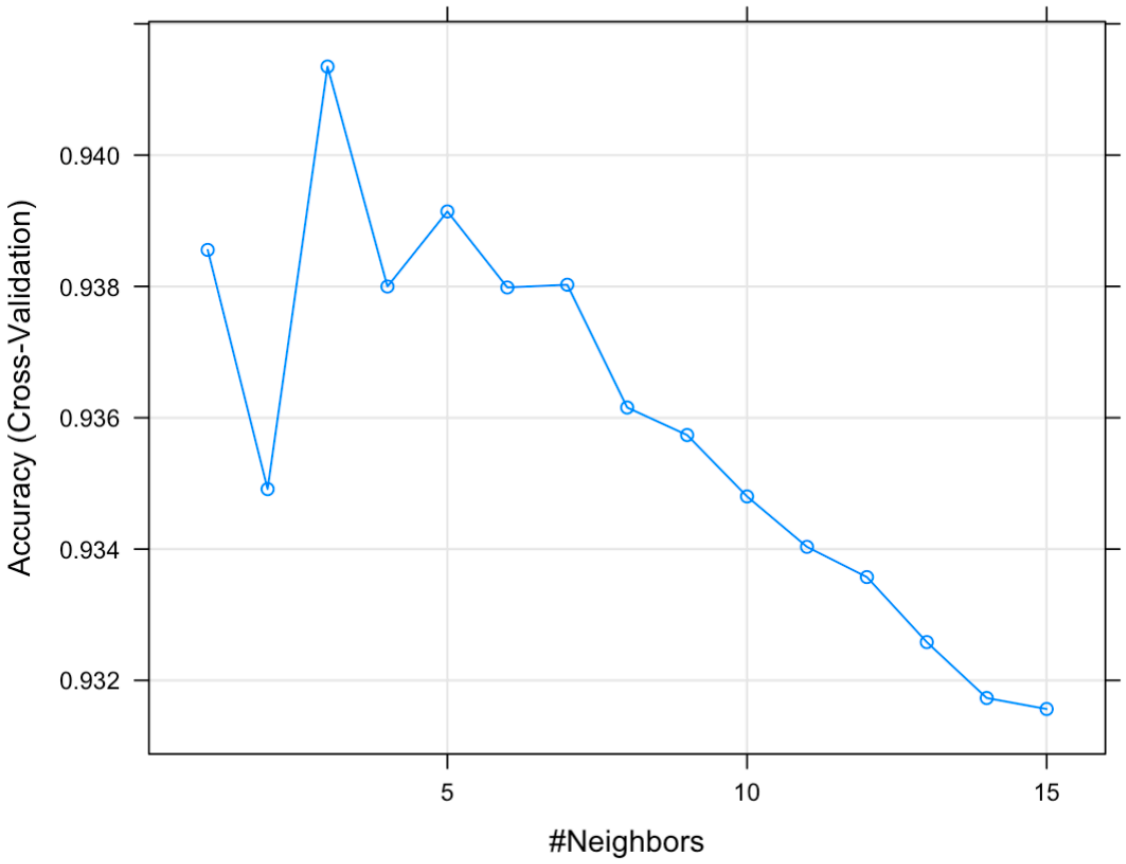
CLASSIFIER 3: K-Nearest Neighbor

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24714	836
Non-Apartment	1276	10081

```
> knn.model4
k-Nearest Neighbors

71643 samples
    7 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64479, 64478, 64479, 64478, 64480, 64479, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9385564  0.8549614
   2  0.9349134  0.8461789
   3  0.9413481  0.8608496
   4  0.9379982  0.8528989
   5  0.9391427  0.8553809
   6  0.9379841  0.8526355
   7  0.9380260  0.8526239
   8  0.9361557  0.8480403
   9  0.9357368  0.8469151
  10  0.9348017  0.8446204
  11  0.9340340  0.8428088
  12  0.9335734  0.8416631
  13  0.9325824  0.8392436
  14  0.9317309  0.8371567
  15  0.9315634  0.8367375

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 3.
```

```
> # Confusion Matrix
> knn.results4 <- confusionMatrix(knn.cm4)
> knn.results4
Confusion Matrix and Statistics

               knn.pred4
                Apartment Non-Apartment
  Apartment         24714           836
  Non-Apartment      1276         10081

               Accuracy : 0.9428
                 95% CI : (0.9404, 0.9451)
    No Information Rate : 0.7042
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8642

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9509
            Specificity : 0.9234
         Pos Pred Value : 0.9673
         Neg Pred Value : 0.8876
             Prevalence : 0.7042
         Detection Rate : 0.6696
   Detection Prevalence : 0.6923
      Balanced Accuracy : 0.9372

       'Positive' Class : Apartment

> # Accuracy
> knn.acc4 <- knn.results4$overall['Accuracy']
> print(knn.acc4)
 Accuracy
0.9427751
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cla
 measures
> knn.metrics4 <- binary_metrics(test4$Property_type, knn.pred4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics4)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9672798 0.11235361 0.9509042 0.9672798 0.9590221 0.9274631 0.8645719
2 Non-Apartment 0.8876464 0.03272016 0.9234222 0.8876464 0.9051809 0.9274631 0.8645719
3  weighted_avg 0.9274631 0.07253689 0.9371632 0.9274631 0.9321015 0.9274631 0.8645719
```


Accuracy (Cross-Validation)
0.940
0.938
0.936
0.934
0.932
5
10
15
#Neighbors

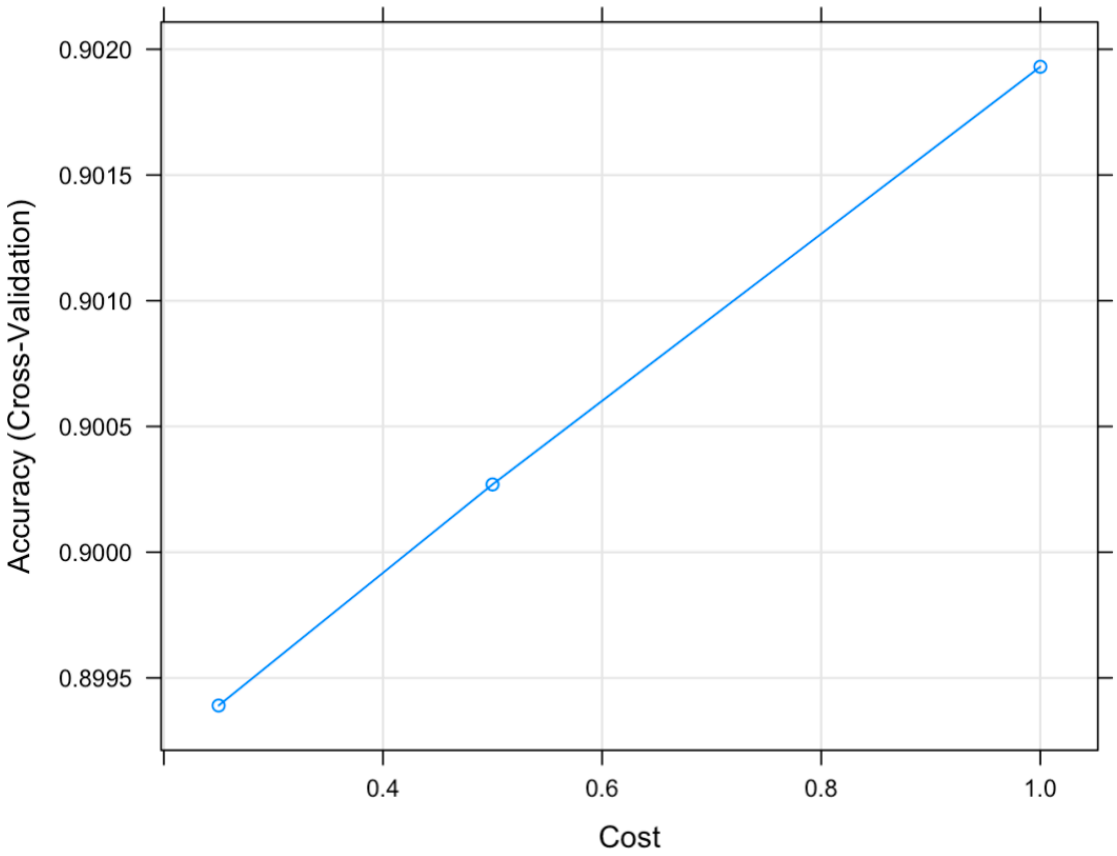
CLASSIFIER 4: Support Vector Machine

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24625	925
Non-Apartment	2593	8764

```
> svm.model4
Support Vector Machines with Radial Basis Function Kernel

71643 samples
    7 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64479, 64479, 64480, 64479, 64478, 64478, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.8993901  0.7534032
  0.50  0.9002695  0.7556897
  1.00  0.9019304  0.7600600

Tuning parameter 'sigma' was held constant at a value of 0.1161237
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1161237 and C = 1.
```

```
> # Confusion Matrix
> svm.results4 <- confusionMatrix(svm.cm4)
> svm.results4
Confusion Matrix and Statistics

               svm.pred4
                Apartment Non-Apartment
  Apartment         24625           925
  Non-Apartment      2593          8764

               Accuracy : 0.9047
                 95% CI : (0.9016, 0.9077)
    No Information Rate : 0.7375
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7668

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9047
            Specificity : 0.9045
         Pos Pred Value : 0.9638
         Neg Pred Value : 0.7717
             Prevalence : 0.7375
         Detection Rate : 0.6672
   Detection Prevalence : 0.6923
      Balanced Accuracy : 0.9046

       'Positive' Class : Apartment

> # Accuracy
> svm.acc4 <- svm.results4$overall['Accuracy']
> print(svm.acc4)
 Accuracy
0.9046793
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cl
 measures
> svm.metrics4 <- binary_metrics(test4$Property_type, svm.pred4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics4)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9637965 0.22831734 0.9047322 0.9637965 0.9333308 0.8677396 0.7714895
2 Non-Apartment 0.7716827 0.03620352 0.9045309 0.7716827 0.8328423 0.8677396 0.7714895
3  weighted_avg 0.8677396 0.13226043 0.9046315 0.8677396 0.8830866 0.8677396 0.7714895
```


Accuracy (Cross-Validation)
0.9020
0.9015
0.9010
0.9005
0.9000
0.8995
0.4
0.6
0.8
1.0
Cost

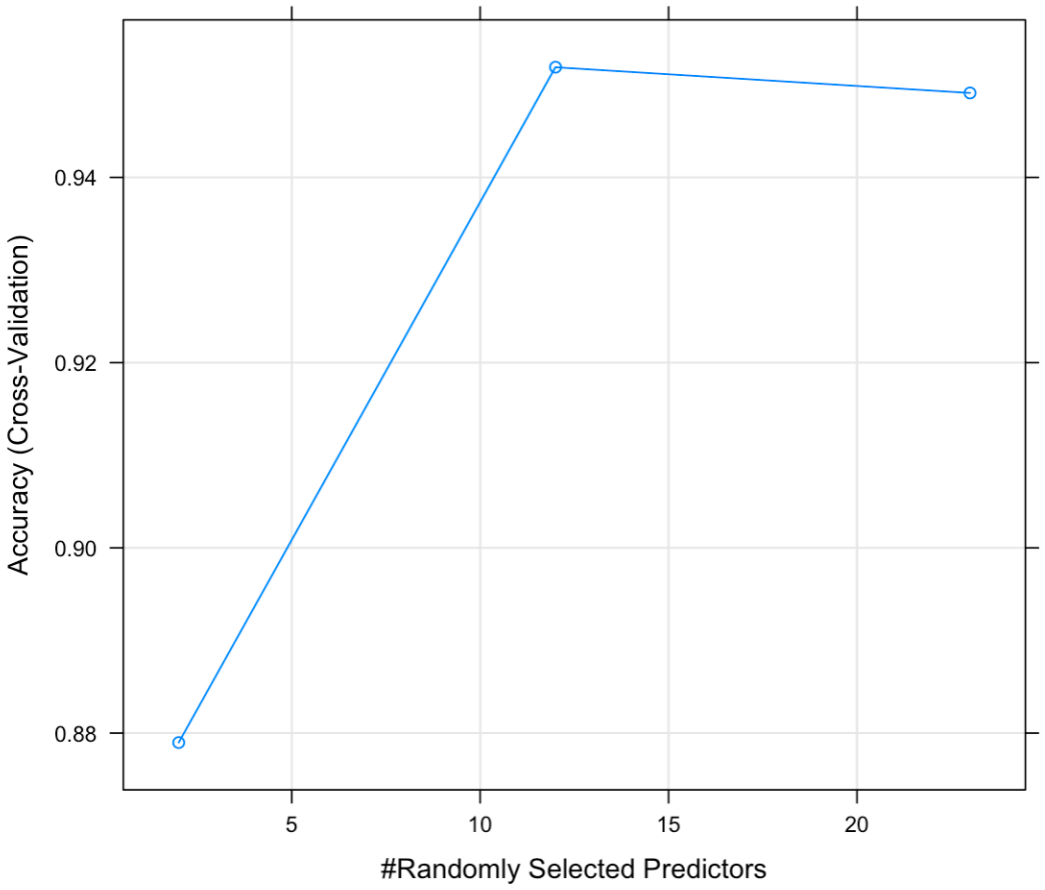
CLASSIFIER 5: Random Forest

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	24816	809
Non-Apartment	959	10401

```
> rf.model4
Random Forest

71795 samples
    7 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 64615, 64616, 64615, 64616, 64615, 64615, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8789749  0.6902823
  12    0.9519048  0.8862740
  23    0.9491190  0.8797017

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 12.
```

```
> # Confusion Matrix
> rf.results4 <- confusionMatrix(rf.cm4)
> rf.results4
Confusion Matrix and Statistics

               rf.pred4
                Apartment Non-Apartment
  Apartment         24816           809
  Non-Apartment       959         10401

               Accuracy : 0.9522
                 95% CI : (0.95, 0.9543)
    No Information Rate : 0.6969
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8873

 Mcnemar's Test P-Value : 0.0003947

            Sensitivity : 0.9628
            Specificity : 0.9278
         Pos Pred Value : 0.9684
         Neg Pred Value : 0.9156
             Prevalence : 0.6969
         Detection Rate : 0.6710
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.9453

       'Positive' Class : Apartment

> # Accuracy
> rf.acc4 <- rf.results4$overall['Accuracy']
> print(rf.acc4)
 Accuracy
0.9521968
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each c
sures
> rf.metrics4 <- binary_metrics(test4$Property_type, rf.pred4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics4)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9684293 0.08441901 0.9627934 0.9684293 0.9656031 0.9420051 0.8873118
2 Non-Apartment 0.9155810 0.03157073 0.9278323 0.9155810 0.9216659 0.9420051 0.8873118
3  weighted_avg 0.9420051 0.05799487 0.9453128 0.9420051 0.9436345 0.9420051 0.8873118
```


Accuracy (Cross-Validation)
0.94
0.92
0.90
0.88
5
10
15
20
#Randomly Selected Predictors

RECURSIVE FEATURE ELIMINATION

```
> print(model)

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD  KappaSD Selected
         0   0.8278 0.5594   0.011327 0.030075
         1   0.8278 0.5594   0.011327 0.030075
         2   0.8536 0.6151   0.008053 0.024922
         3   0.8608 0.6347   0.009487 0.030433
         4   0.8850 0.7143   0.008376 0.024847
         5   0.8920 0.7361   0.006924 0.018375
         6   0.9246 0.8193   0.005172 0.012981
         7   0.9339 0.8424   0.004401 0.010146
         8   0.9351 0.8454   0.004286 0.010373
         9   0.9380 0.8523   0.004068 0.009154
        10   0.9402 0.8576   0.004593 0.010547        *
        11   0.9393 0.8553   0.004956 0.011406
        12   0.9384 0.8534   0.005223 0.012333

The top 5 variables (out of 10):
   No_of_BHK, Property_building_status, Price_per_unit_area, Size, Price

> # get selected features
> selected_features <- model$optVariables[1:5]; selected_features
[1] "No_of_BHK"                "Property_building_status" "Price_per_unit_area"
[4] "Size"                     "Price"
```

CLASSIFIER	ACCURACY
GLM (Logistic Regression)	84.26%
Naive Bayes	72.18%
k-NN	88.46%
Support Vector Machine	87.63%
Random Forest	88.94%

CLASSIFIER 1: GLM Model (Logistic regression)

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	3515	210
Non-Apartment	634	1005

```
> log.model5
Generalized Linear Model

10413 samples
    5 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9371, 9372, 9372, 9372, 9372, 9371, ...
Resampling results:

  Accuracy   Kappa
  0.8512452  0.6255646
```

```
> log.results5
Confusion Matrix and Statistics

               log.pred5
                Apartment Non-Apartment
  Apartment          3515           210
  Non-Apartment       634          1005

               Accuracy : 0.8427
                 95% CI : (0.8326, 0.8523)
    No Information Rate : 0.7735
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6003

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8472
            Specificity : 0.8272
         Pos Pred Value : 0.9436
         Neg Pred Value : 0.6132
             Prevalence : 0.7735
         Detection Rate : 0.6553
   Detection Prevalence : 0.6944
      Balanced Accuracy : 0.8372

       'Positive' Class : Apartment

> # Accuracy
> log.acc5 <- log.results5$overall['Accuracy']
> print(log.acc5)
 Accuracy
0.8426547
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each class
measures
> log.metrics5 <- binary_metrics(test5$Property_type, log.pred5)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(log.metrics5)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9436242 0.38682123 0.8471921 0.9436242 0.8928118 0.7784015 0.6127654
2 Non-Apartment 0.6131788 0.05637584 0.8271605 0.6131788 0.7042747 0.7784015 0.6127654
3  weighted_avg 0.7784015 0.22159854 0.8371763 0.7784015 0.7985432 0.7784015 0.6127654
> # Define the cross validation method
```

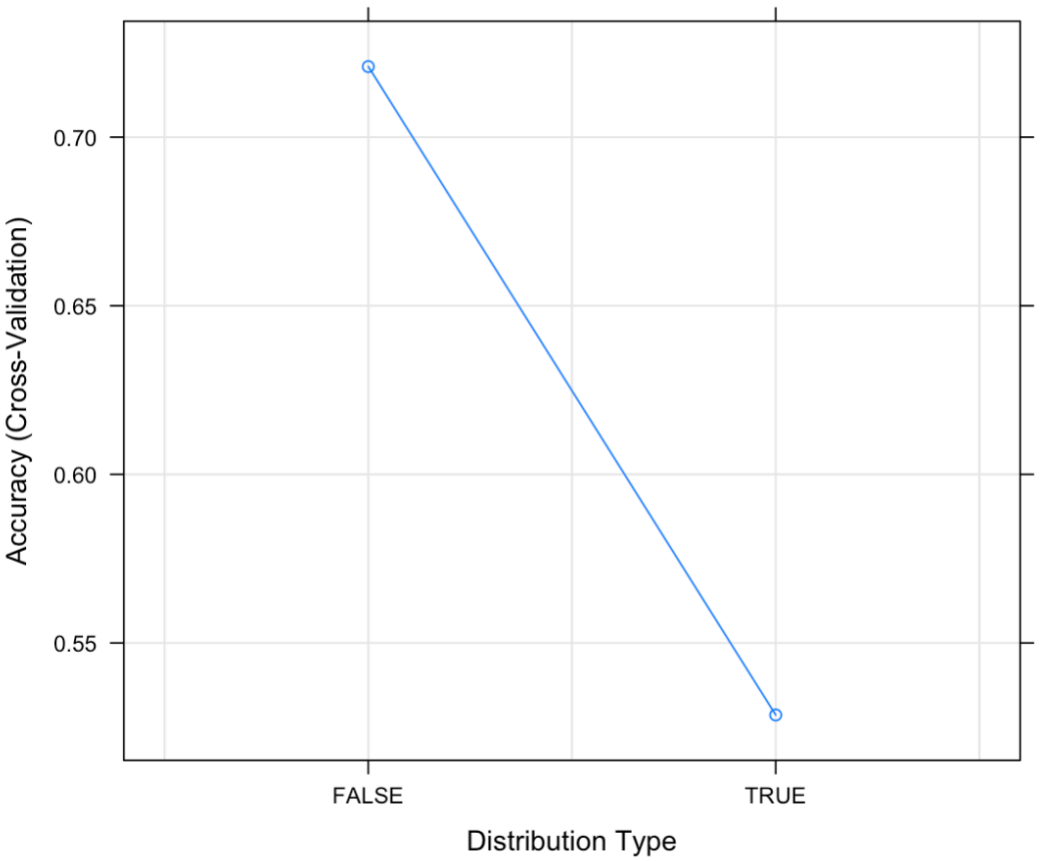
CLASSIFIER 2: Naive Bayes

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	3675	50
Non-Apartment	1442	197

```
> n.model5
Naive Bayes

10413 samples
    5 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9372, 9372, 9372, 9372, 9371, 9371, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7214056  0.1372680
   TRUE      0.5172270  0.2167807

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = FALSE and adjust = 1.
```

```
> n.results5
Confusion Matrix and Statistics

               n.pred5
                Apartment Non-Apartment
  Apartment          3675            50
  Non-Apartment      1442           197

               Accuracy : 0.7218
                 95% CI : (0.7096, 0.7338)
    No Information Rate : 0.954
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1401

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7182
            Specificity : 0.7976
         Pos Pred Value : 0.9866
         Neg Pred Value : 0.1202
             Prevalence : 0.9540
         Detection Rate : 0.6851
   Detection Prevalence : 0.6944
      Balanced Accuracy : 0.7579

       'Positive' Class : Apartment

> # Accuracy
> n.acc5 <- n.results5$overall['Accuracy']
> print(n.acc5)
 Accuracy
0.7218494
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each c
measures
> n.metrics5 <- binary_metrics(test5$Property_type, n.pred5)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(n.metrics5)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9865772 0.87980476 0.7181943 0.9865772 0.8312599 0.5533862 0.2346689
2 Non-Apartment 0.1201952 0.01342282 0.7975709 0.1201952 0.2089077 0.5533862 0.2346689
3  weighted_avg 0.5533862 0.44661379 0.7578826 0.5533862 0.5200838 0.5533862 0.2346689
```


Accuracy (Cross-Validation)
0.70
0.65
0.60
0.55
FALSE
TRUE
Distribution Type

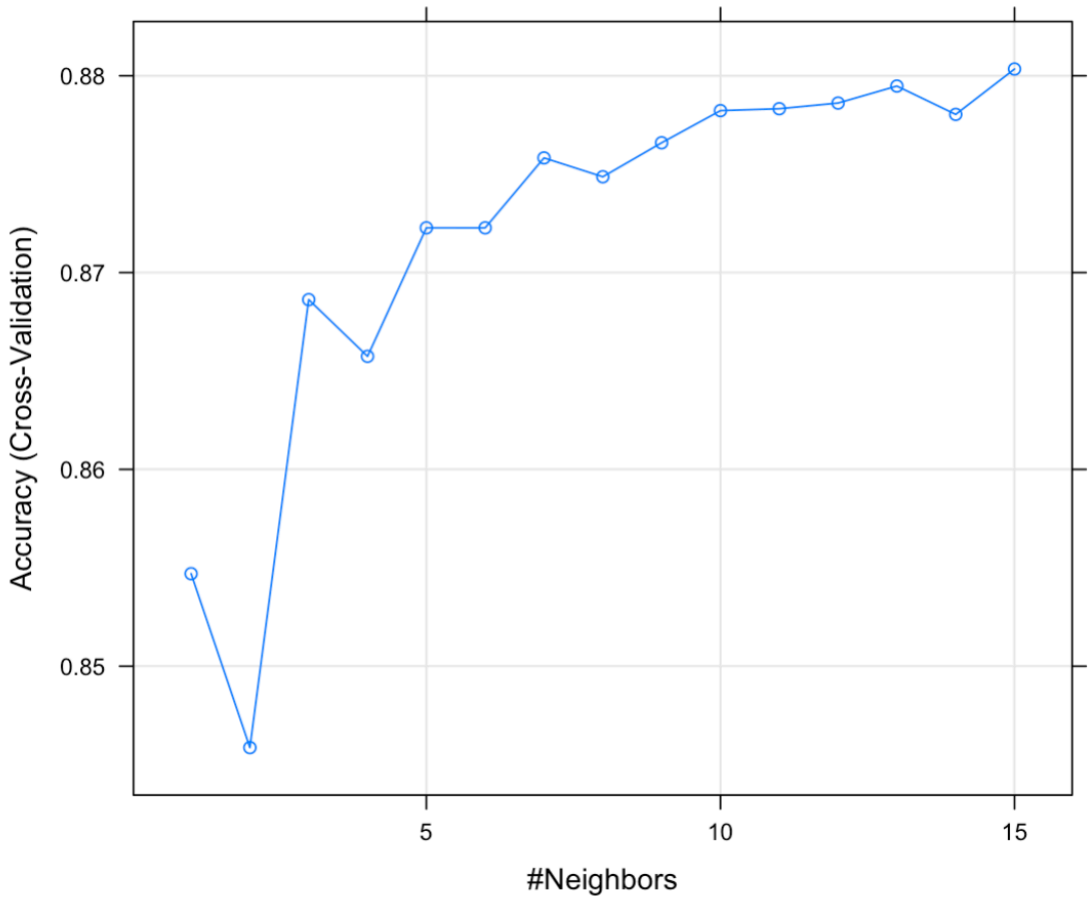
CLASSIFIER 3: K-Nearest Neighbor

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	3539	186
Non-Apartment	433	1206

```
> knn.model5
k-Nearest Neighbors

10413 samples
    5 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9372, 9372, 9371, 9372, 9372, 9372, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8547001  0.6583380
   2  0.8458641  0.6380597
   3  0.8686243  0.6856777
   4  0.8657441  0.6784341
   5  0.8722745  0.6916631
   6  0.8722738  0.6915008
   7  0.8758277  0.6987257
   8  0.8748680  0.6954698
   9  0.8765970  0.6991802
  10  0.8782291  0.7028487
  11  0.8783257  0.7024716
  12  0.8786137  0.7027608
  13  0.8794780  0.7046843
  14  0.8780375  0.7010369
  15  0.8803427  0.7064965

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
```

```
> # Confusion Matrix
> knn.results5 <- confusionMatrix(knn.cm5)
> knn.results5
Confusion Matrix and Statistics

               knn.pred5
                Apartment Non-Apartment
  Apartment          3539           186
  Non-Apartment       433          1206

               Accuracy : 0.8846
                 95% CI : (0.8757, 0.893)
    No Information Rate : 0.7405
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7161

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8910
            Specificity : 0.8664
         Pos Pred Value : 0.9501
         Neg Pred Value : 0.7358
             Prevalence : 0.7405
         Detection Rate : 0.6598
   Detection Prevalence : 0.6944
      Balanced Accuracy : 0.8787

       'Positive' Class : Apartment

> # Accuracy
> knn.acc5 <- knn.results5$overall['Accuracy']
> print(knn.acc5)
Accuracy
0.884601
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cl
measures
> knn.metrics5 <- binary_metrics(test5$Property_type, knn.pred5)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(knn.metrics5)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9500671 0.26418548 0.8909869 0.9500671 0.9195791 0.8429408 0.7207382
2 Non-Apartment 0.7358145 0.04993289 0.8663793 0.7358145 0.7957770 0.8429408 0.7207382
3  weighted_avg 0.8429408 0.15705918 0.8786831 0.8429408 0.8576780 0.8429408 0.7207382
```


Accuracy (Cross-Validation)
0.88
0.87
0.86
0.85
5
10
15
#Neighbors

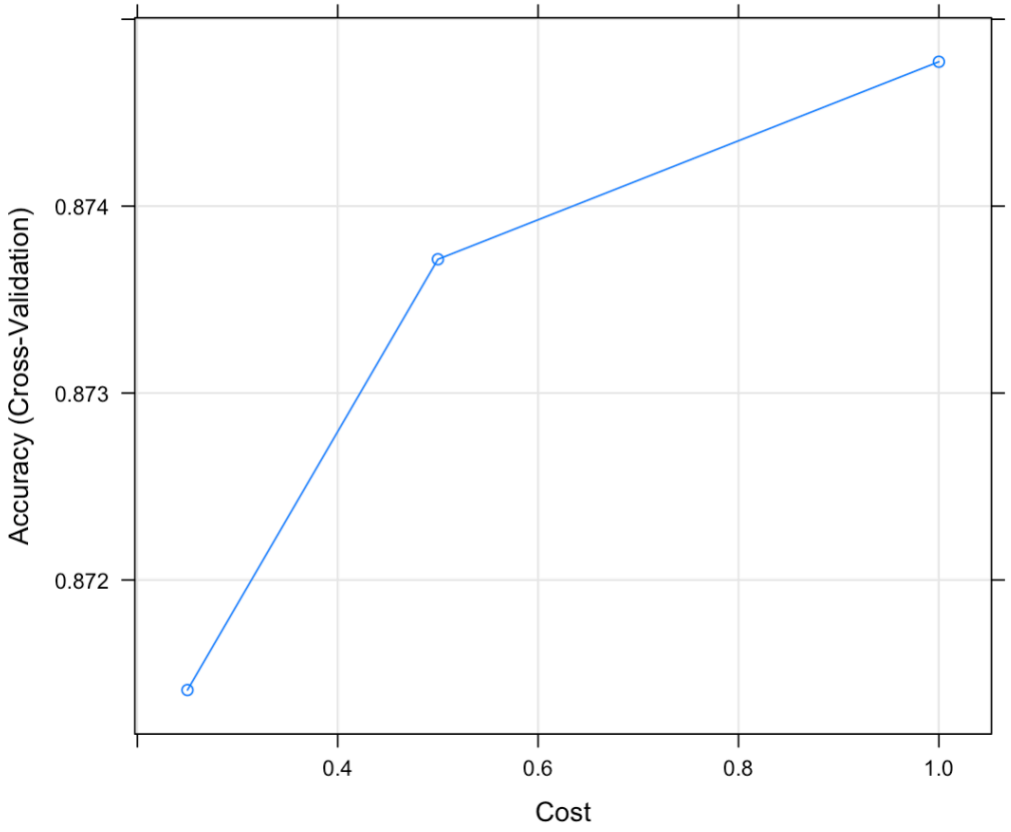
CLASSIFIER 4: Support Vector Machine

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	3599	126
Non-Apartment	537	1102

```
> svm.model5
Support Vector Machines with Radial Basis Function Kernel

10413 samples
    5 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9372, 9372, 9371, 9372, 9372, 9372, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.8714111  0.6711589
  0.50  0.8737157  0.6778785
  1.00  0.8747723  0.6806090

Tuning parameter 'sigma' was held constant at a value of 0.3793572
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.3793572 and C = 1.
```

```
> svm.results5 <- confusionMatrix(svm.cm5); svm.results5
Confusion Matrix and Statistics

               svm.pred5
                Apartment Non-Apartment
  Apartment          3599           126
  Non-Apartment       537          1102

               Accuracy : 0.8764
                 95% CI : (0.8673, 0.8851)
    No Information Rate : 0.7711
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6868

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8702
            Specificity : 0.8974
         Pos Pred Value : 0.9662
         Neg Pred Value : 0.6724
             Prevalence : 0.7711
         Detection Rate : 0.6710
   Detection Prevalence : 0.6944
      Balanced Accuracy : 0.8838

       'Positive' Class : Apartment

> # Accuracy
> svm.acc5 <- svm.results5$overall['Accuracy']
> print(svm.acc5)
 Accuracy
0.8763982
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each c
measures
> svm.metrics5 <- binary_metrics(test5$Property_type, svm.pred5)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(svm.metrics5)
          class   TP_rate   FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9661745 0.3276388 0.8701644 0.9661745 0.9156596 0.8192678 0.7000811
2 Non-Apartment 0.6723612 0.0338255 0.8973941 0.6723612 0.7687478 0.8192678 0.7000811
3  weighted_avg 0.8192678 0.1807322 0.8837793 0.8192678 0.8422037 0.8192678 0.7000811
```


Accuracy (Cross-Validation)
0.874
0.873
0.872
0.4
0.6
0.8
1.0
Cost

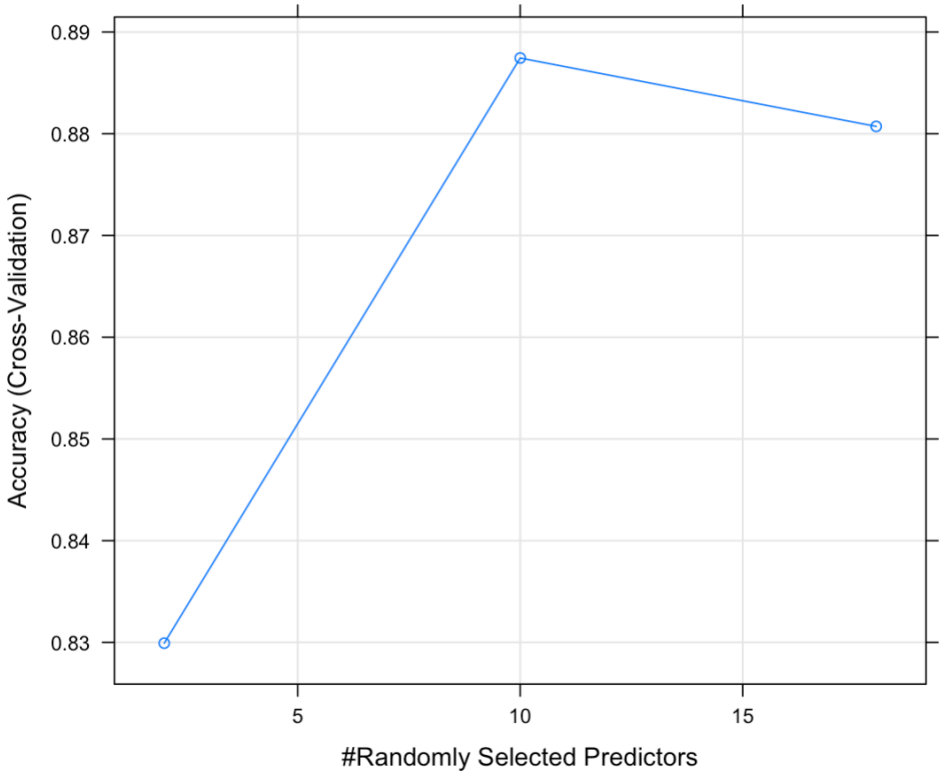
CLASSIFIER 5: Random Forest

Confusion Matrix:

	Apartment	Non-Apartment
Apartment	3531	194
Non-Apartment	399	1240

```
> rf.model5
Random Forest

10413 samples
    5 predictor
    2 classes: 'Apartment', 'Non-Apartment'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9372, 9371, 9371, 9372, 9371, 9372, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8299238  0.5390865
  10    0.8874464  0.7242486
  18    0.8807241  0.7143789

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was mtry = 10.
```

```
> rf.results5
Confusion Matrix and Statistics

               rf.pred5
                Apartment Non-Apartment
  Apartment          3531           194
  Non-Apartment       399          1240

               Accuracy : 0.8894
                 95% CI : (0.8808, 0.8977)
    No Information Rate : 0.7327
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.73

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8985
            Specificity : 0.8647
         Pos Pred Value : 0.9479
         Neg Pred Value : 0.7566
             Prevalence : 0.7327
         Detection Rate : 0.6583
   Detection Prevalence : 0.6944
      Balanced Accuracy : 0.8816

       'Positive' Class : Apartment

> # Accuracy
> rf.acc5 <- rf.results5$overall['Accuracy']
> print(rf.acc5)
 Accuracy
0.8894482
> # Get the TP rate, FP rate, precision, recall, F-measure, ROC area, and MCC for each cl
measures
> rf.metrics5 <- binary_metrics(test5$Property_type, rf.pred5)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(rf.metrics5)
          class   TP_rate    FP_rate precision    recall F_measure  ROC_area       MCC
1     Apartment 0.9479195 0.24344112 0.8984733 0.9479195 0.9225343 0.8522392 0.7332455
2 Non-Apartment 0.7565589 0.05208054 0.8647141 0.7565589 0.8070290 0.8522392 0.7332455
3  weighted_avg 0.8522392 0.14776083 0.8815937 0.8522392 0.8647816 0.8522392 0.7332455
```


0.89
0.88
0.87
0.86
0.85
0.84
0.83
Accuracy (Cross-Validation)
5
10
15
#Randomly Selected Predictors

OVERALL RESULTS

Accuracy

Model / Method	Random Forest Feature Selection	Boruta Model	Chi Square	R-Part	Recursive Feature Elimination
Logistic Regression	83.63%	83.49%	83.49%	87.66%	84.26%
Naive Bayes	79.25%	73.64%	73.64%	86.00%	72.18%
K-Nearest Neighbors	85.72%	85.52%	85.52%	94.27%	88.46%
Support Vector Machine	84.73%	84.87%	84.87%	90.46%	87.63%
Random Forest	87.05%	95.51%	87.11%	95.21%	88.94%

Precision

Model / Method	Random Forest Feature Selection	Boruta Model	Chi Square	R-Part	Recursive Feature Elimination
Logistic Regression	0.8639	0.8871	0.8607	0.8726	0.8371
Naive Bayes	0.7569	0.8291	0.8170	0.8326	0.7578
K-Nearest Neighbors	0.8489	0.9324	0.8458	0.9371	0.8786
Support Vector Machine	0.8699	0.8973	0.8639	0.9046	0.8837
Random Forest	0.8756	0.9498	0.8765	0.9453	0.8816

Recall

Model / Method	Random Forest Feature Selection	Boruta Model	Chi Square	R-Part	Recursive Feature Elimination
Logistic Regression	0.7495	0.8619	0.7473	0.8305	0.7781
Naive Bayes	0.7534	0.8493	0.5753	0.8648	0.5533
K-Nearest Neighbors	0.8061	0.9235	0.8032	0.9274	0.8429
Support Vector Machine	0.7687	0.8795	0.7738	0.8677	0.8192
Random Forest	0.8131	0.9445	0.8138	0.9420	0.8522

CONCLUSION

Best Feature Selection Method:
Based on the tables, it seems that the Boruta feature selection method generally performs the best across the different models and evaluation metrics. The Boruta model has the highest precision and recall scores for most models. It also has the highest accuracy scores for the Naive Bayes, Support Vector Machine, and Random Forest models in the accuracy table. However, Boruta model is very susceptible to overfitting and this might be one of the cases the accuracy is so high. We can also observe that Random Forest Feature Selection is quite consistent across all models and so it can be considered as one of the best feature selection methods.

Best Model:
Looking at the average accuracy, precision, and recall values for each model, we can see that the Random Forest model consistently performs well across all attribute selection methods, followed by the K-Nearest Neighbors and Support Vector Machine models. The Logistic Regression and Naive Bayes models generally have lower average performance across all attribute selection methods. Therefore, we can conclude that the Random Forest model is the best performing model across all classifiers.

Project Learnings:
- Feature selection is a crucial step in machine learning, and relevant feature selection techniques can reduce the time taken to train models.
- The accuracy of machine learning classifiers is highly dependent on the quality and quantity of the data used to train them.
- The project improved proficiency in training machine learning models using the R programming language.
- Gained hands-on experience in data preprocessing, feature selection, model selection, and evaluation.
- Gained a deeper understanding of the strengths and limitations of different classification models and how to optimize their performance.
- Experience could be useful for future machine learning projects in various domains.

CONTRIBUTION

Aditya Maheshwari:
- Getting the Data
- Preprocessing the Data
- Found the 5 attribute selection methods
- Found the 2 classifiers that suit the use-case
- Trained half the models
- Wrote the report

Vaidehi Shah:
- Cleaning the Dataset
- Created the performance metric custom function.
- Found the remaining 3 classifiers that suit the use-case
- Trained half the models
- Wrote the report

APPENDIX

A. Dataset Source and Code Link:
https://drive.google.com/drive/folders/19Rpme2kL-1SfO1gJJhei1tDtmwYnglBG?usp=share_link