# INF 552: Machine Learning for Data Informatics
# Click Through Rate Prediction

## Problem Definition:

In online advertising, click-through rate (CTR) is a very important metric for evaluating ad performance. As a result, click prediction systems are essential and widely used for sponsored search and real-time bidding. Hence, we want to implement using machine learning algorithms and pick the one, which does the best job.

## Background :

We want to predict whether an advertisement (ad) will be clicked based on the feature values like website-id and app-id in the dataset. As this is a classification problem, we want to use supervised learning algorithms taught in the class and some algorithms, which are under research. Since we don't have any information about the data distribution, we had taken non-parametric Machine learning algorithms. In this project we want to predict the value based on high accuracy by training Random Forest Algorithm, Multi-Layer Perceptron and Support Vector Machine.

## Dataset Description:

We have taken dataset from Avazu (Kaggle)[1], which contains 11 days of data to build and test various machine-learning algorithms. As given data is approximately 6GB, we have decided to take only 10 hrs of data for training (150 MB), 2 hrs. for testing (30 MB). We used "hour" feature in the dataset, which gives day and time at which data is recorded, we picked the data for the first day using this feature value and sampled the first day's data so that we get a uniform distribution.

- **Train** - Training set. 10 hours of click-through data, ordered chronologically. Non-clicks and clicks are subsampled according to different strategies.
- **Test** - Test set. 2 hours of ads to for testing our model predictions.

## Features:

- id: advertisement identifier which is unique to every advertisement
- click: 0/1 for non-click/click - which is the label we have to predict
- hour: format is YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014
- C1 -- anonymized categorical variable - number
- banner_pos -- 0 for top , 1 for bottom - number
- site_id - website identifier - string
- site_domain - website domain id - string
- site_category - category to which website belongs to - string

- app_id - application identifier - string
- app_domain - application domain - string
- app_category - category to which application belongs to - string
- device_id - device from which ad was clicked - string
- device_ip – device ip address - string
- device_model - device model number - string
- device_type - device– mobile, desktop or tablet – integer - categorical
- device_conn_type – internet connection type – wifi, mobile data - categorical
- C14-C21 – another set of anonymized categorical variables

**Tools:**

**Scikit :**

As python is open source and emerging popularly as one of the best toolbox in the field of machine learning we have planned to use scikit, python library which is built on scipy, numpy and matplotlib.

**Pandas:**

Python has long been great for data munching and preparation, but less so for data analysis and modeling. Pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.

**PyBrain:**

PyBrain is the abbreviation for Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library. As the name suggests PyBrain is a powerful tool for implementing neural network. PyBrain is also flexible with the feature for simple and complex implementation.

**Related work:**

- Random Forests - http://goo.gl/vFxidb
- SVM - http://goo.gl/3SNdif
- Neural Networks - http://goo.gl/rw1tXC

**Analysis of features:**

We used unique() function in scikit to find the categorical variable values

We used boxplot to visualize the feature values for numerical values and histograms for categorical values.

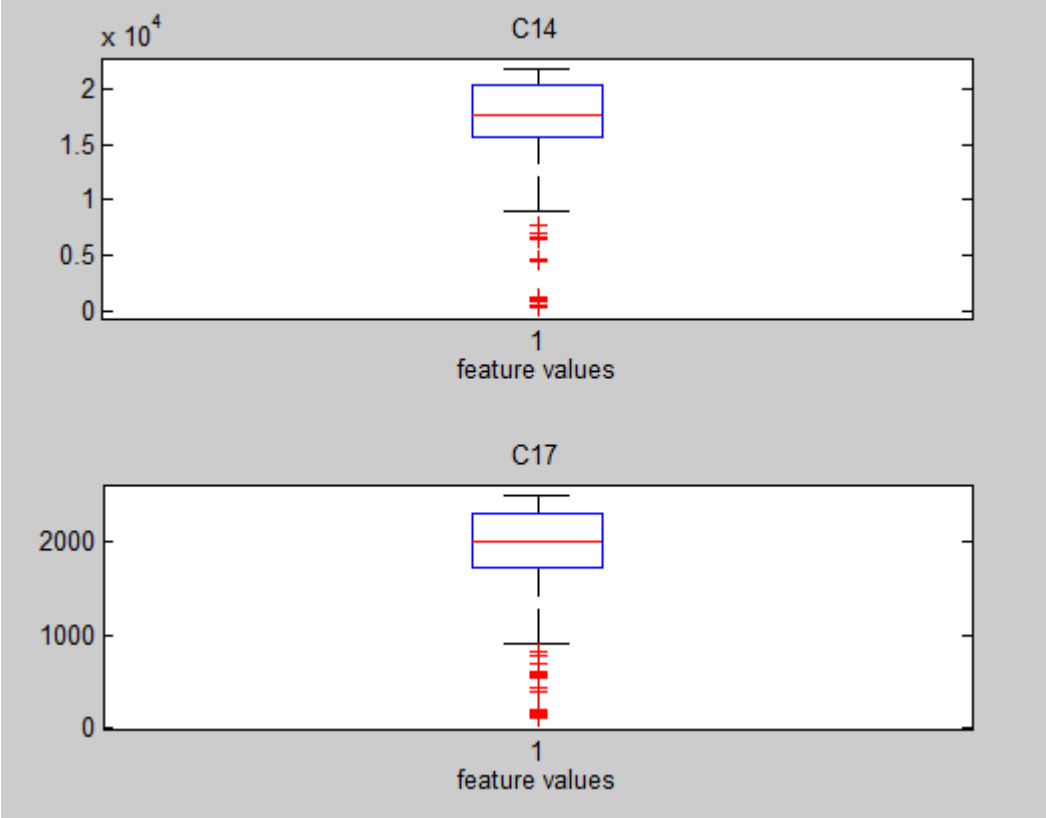


*Fig 1.1: Histogram for C14 and C17. Since these are the numerical features in the dataset. The same can be done for other feature set.*

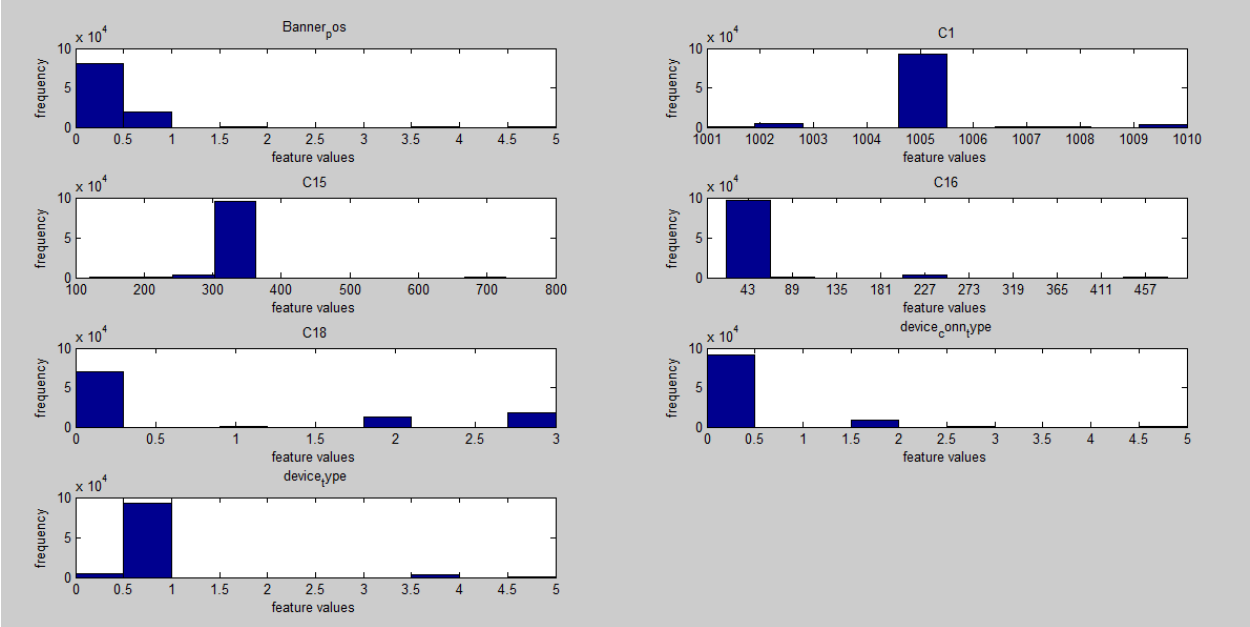


*Fig 1.2: Histograms for certain features(from left to right) (1).Banner_pos – {0 , 1, 4, 5} (2) C1 –{1001, 1002,1005,1007,1008, 1010 } (3) C15– {216, 300, 320, 728}*
*(4) C16 {36, 50, 90, 250, 480} (5) C18 {0, 1, 2 3} (6) device_conn_type { 0, 2, 3, 5} (7) device_type {0, 1, 4, 5}*

**PCA Analysis :**

We calculated feature relevance to target variable click using PCA and the values look like

```
[ 2.80294829e-01  1.67424427e-01  1.61079887e-01  1.56639544e-01
  8.37471324e-02  5.05418560e-02  4.30371006e-02  2.56391464e-02
  1.64314095e-02  1.51646679e-02  1.26141436e-29  4.48414245e-32
  6.50877800e-33  1.53135799e-33  3.38620537e-34  2.86302692e-34
  2.33262750e-34  1.30853826e-34  4.53292849e-35  1.53253332e-35
  8.83786438e-36  1.93075657e-36  0.00000000e+00]
```

We can observe that the first 10 features have less variance, so they have most influence on the target variable.
The one with the higher variance has large correlation with the target variable, so in our experiment we tried reducing the number of features from 23 to 15 and 10 and evaluated the results.

**Experiment:**

We conducted Machine learning experiment by implementing the following phases.

1. **Preprocessing**

   The dataset downloaded from AVAZU website consists of both categorical and continuous variables in integers and string format. Since Decision trees implementation of scikit supports for features of type integer only, we had to pre-process dataset in-order to convert string variables to integers. We used pandas library to load csv file to memory, identifies the columns with string values and converted to integer values using python standard hash function.

2. **Machine Learning Algorithm**

   We have used Random Forest,Neural Networks and Support Vector Machine in our project. We used scikit to implement Random Forest classifier, SVM and MLP.

   **Random Forest:**

   - A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.
   - We used RandomForestClassifier()[3] from scikit library to perform classification
   - We used RandomForestClassifier().fit(features,target)[3] to train the given model

.

**Results for Random Forest**:

| Number of trees | Features used | Accuracy | F-measure |
|---|---|---|---|
| 100 | Auto | 83.19 | 90.5,17.5 |
| 100 | Log2 | 83.19 | 90.9,21.2 |
| 300 | Auto | 83.39 | 90.6, 19.3 |
| 300 | Log2 | 82.89 | 90.3, 18.1 |
| 500 | Auto | 83.19 | 90.5, 18.3 |
| 500 | Log2 | 83.29 | 90.6, 18.44 |

*Table 1: Observations for random forest by changing parameters like number of trees used and max features used*

By observing the above results we can justify that Random Forest is a good choice because it's showing a consistent accuracy of 83 %

**Neural Networks:**

- Artificial neural networks used for this experiment consists of three layers. Input layer, hidden layer and output layer. The neural network has to predict whether the user will click the ad or not so basically the experiment is a binary classification. Experiment is conducted by varying the number of neurons in the hidden layer and the number of epoch [6].
- The results for the experiment are given in the table below.

| Epoch number | Number of neurons in hidden layer | Accuracy | Precision | Recall | F-score |
|---|---|---|---|---|---|
| 1 | 200 | 55.2 | 79.3,19.1 | 59.5,38.2 | 68.2,25.3 |
| **2** | **200** | **83.2** | **83.8,0.2** | **99.04,0.2** | **90.8, 0.1** |
| 3 | 200 | 81.2 | 82.6,0.37 | 97.5,0.06 | 89.4,0.1 |
| 1 | 400 | 30.1 | 90.3, 14.1 | 21.6, 84.8 | 34.9, 24.2 |
| 2 | 400 | 44.4 | 81.8,0.15 | 43.06,0.51 | 56.4,0.23 |
| 1 | 300 | 50.4 | 80.4, 10.2 | 54.5, 28.2 | 64.97, 15.06 |
| 2 | 300 | 76.8 | 81.1,0.17 | 93.1,0.06 | 86.6,0.09 |
| 1 | 500 | 35.6 | 75.32, 17.9 | 29, 62 | 41.8, 0.21 |
| 2 | 500 | 48.4 | 88.6,0.18 | 44.5,0.69 | 59.3,0.29 |
| 1 | 600 | 19.6 | 0, 19.6 | 0, 100 | 0.1, 32.77 |
| 2 | 600 | 50 | 93,0.23 | 42.5,0.8 | 58.4,0.37 |

*Table 2: Observations for neural network by changing parameters like number of neurons in hidden layer*

**Observation:** Increasing the hidden neurons and epoch number beyond certain extent causes over fitting in the training data and hence degrades the accuracy

**Support Vector Machine**:

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier detection.

**Advantages:**

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel Functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

**Disadvantages:**

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

We varied the C parameter from $10^{-5}$ to $10^{4}$ and computed accuracy using 10-fold cross-validation [5].

| **C value** | **Accuracy** |
|---|---|
| 0.00001 | 61.38 |
| 0.0001 | 61.38 |
| 0.001 | 61.38 |
| 0.01 | 61.38 |
| 0.1 | 61.38 |
| 1 | 61.38 |
| 10 | 61.38 |
| 100 | 61.38 |

| | |
|---|---|
| 1000 | 61.38 |
| 10000 | 61.38 |

*Table 3: Observations for SVM by changing C value*

After cross-validation, we chose C = $10^{-5}$ and trained SVM using 100,000 records (30 MB). We predicted the algorithm in 1000 records and computed confusion matrix, precision, recall and accuracy parameters.

### 3. Train the model using training dataset

We have taken a training dataset of about 100 thousand rows which corresponds to of size 30MB(which gives click through rate of 10 hours in a day ). We first pre-processed this dataset as mentioned above and created a new dataset file which can be act as the normalized dataset. Later, this dataset is used for all the algorithms which are mentioned previously

### 4. Evaluate the model with test data

We have taken 1% of total size of training dataset (1 hour in a day) as test data which contains 1000 records which are not seen during training phase.

## Evaluation Metrics:

| Classifier | Accuracy | Confusion Matrix | Precision | Recall | F-measure |
|---|---|---|---|---|---|
| Neural Networks (neurons = 200, 1 layer) | 82.68 % | [ 816 14<br>159 10 ] | 83.8,0.2 | 99.04,0.2 | 90.8, 0.1 |
| SVM ( C = 0.00001, kernel='rbf') | 66.7 % | [ 831 0<br>169 0] | 82.8, 0.2 | 99.5, 4.5 | 90.5, 2.6 |
| Random Forest(n-estimators = 100) | 83.1 % | [ 815 16<br>151 18] | 84.1,10.3 | 97.5,2.1 | 91.2, 3.2 |

*Table 1: Comparison of 3 algorithms over a training set of 1 Lakh rows and test dataset of 1000 rows*

Confusion matrix tells how many 0s are predicted and how many of them are actually zeros and ones. For example, confusion matrix of neural networks out of 830 0s, 816 are predicted correctly.

From the above numbers we can observe that Random Forest and Neural Network shows an accuracy of 83 %. But SVM shows an accuracy of 66.7% is not a good choice because the confusion matrix of SVM shows that its always predicts 0.

## 5. Experiment Visualization:

### ROC Curve:

ROC Curve plots True Positive Rate vs False Positive Rate. This is a good visualization for comparing various algorithms on a single plot. Here, we will show the ROC Curve for the 3 algorithms. The area covered by the graph depicts the number of classifications it made correctly.

After performing validation using test dataset, we saved both predicted and actual label for each algorithm. We created a python script which will read all these files and plotted ROC curve in same graph in-order to get better visualization about the performance and evaluation results.

This proves us that Random Forest performs better than other 2 algorithms by covering large area.

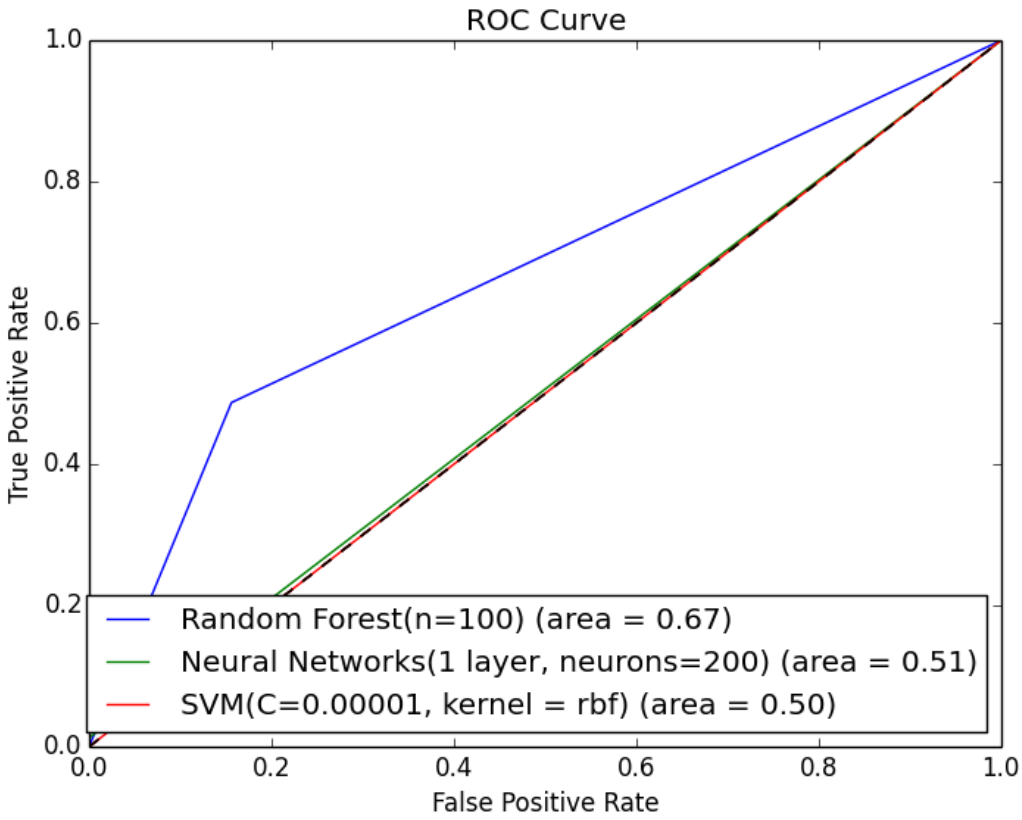


*Fig 3: ROC curve for SVM, Neural Networks and Random Forest algorithms*

## Conclusion:

Random forest uses a bagging method for classification. From the ROC Curve, we can infer that Random Forest is performing much better than other algorithms. The more area an algorithm covers on the graph the better it is. As Random Forest with n = 100 covers area of 0.67 this is better than other algorithms. Also, Random forest considers a number of decision trees, so it's robust to overfitting and gives a better accuracy. We can train the Random Forest to run using complete dataset ( 6 GB ) in a distributed environment either by using Apache Mahout or Apache Spark Matlin, but by sampling the dataset with equal probability , we could able to achieve the desired results by executing in a single machine.

## How related work is used in the experiment:

**Random Forest[2]:**

In the technical paper[2] author discusses about the feature selection and how various number of tree nodes results in different tree with different structure, which helps to determine best structure among the different tree. Also, the author conducted experiment result with different number of nodes by varying tree nodes.

**Support Vector Machines [4]:**

In the technical paper [4], author suggested various methodologies to find the optimal or nearest optimal parameter values involved in SVM experiment. His main candidate to find out 'C' value is using cross-validation or leave-oneout method, as the dataset can perform poor and overfit by considering leave-one out method, we decided to perform cross-validation.

**Artificial Neural Networks[7]:**

ANN experiment is conducted based on the concepts described in the paper [7]. Metrics for the experiment are calculated by varying the number of hidden layer perceptrons and the epoch number. Optimal number of epoch and perceptrons yielded good results. Poor choice lead to over fitting.

## References:


[1] https://www.kaggle.com/c/avazu-ctr-prediction
[2] Random Forests - http://goo.gl/vFxidb
[3] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble RandomForestClassifier.html
[4] SVM - http://goo.gl/3SNdif
[5] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] http://pybrain.org/docs/tutorial/fnn.html
[7] http://goo.gl/rw1tXC


**Source Code**:

The complete source code is attached as zip file in the blackboard.