CS 521-DATA MINING PROJECT FINAL REPORT

KAGGLE COMPETETION

LIBERTY MUTUAL INSURANCE-Quantify property hazards before time of inspection

MAGESH RAJASEKARAN
SRI RANJITHA SANKAR
THILAK RAJ BALASUBRAMANIAN

PROBLEM DEFNITION:

Liberty Mutual's Home Insurance:
To ensure that Liberty Mutual's portfolio of home insurance policies aligns with their business goals, many newly insured properties receive a home inspection.
These inspections review the condition of key attributes of the property, including things like the foundation, roof, windows and siding. The results of an inspection help Liberty Mutual determine if the property is one they want to insure.

In this Kaggle Competition we should predict a transformed count of hazards or pre-existing damages using a dataset of property information. This will enable Liberty Mutual to more accurately identify high risk homes that require additional examination to confirm their insurability.

This is a Prediction problem to predict the Hazard score of a particular property given with its attributes. We need to build a prediction model from the given training set and predict the Hazard count of the Properties in Test Dataset.

What is **Numeric Prediction**? Prediction is similar to classification, constructs a model and uses the model to predict unknown or missing values and the Predicted values are usually continuous whereas classifications are discreet. [1]
We are going to predict the Hazard score of each property using the model built by the train data of huge list of properties along with its features and Hazard score. Hence it is going to be **Supervised Learning Algorithm** approach.

DATA DESCRIPTION:

Provided with csv files **train.csv** indicating train data and **test.csv** indicating test data for which we need to predict the hazard score and submit the hazard scores of the Test data in the prescribed csv format (from **sample_submission.csv**) to kaggle to view the status and leader board score.

Train.csv	51000 data	32 features(columns)
Test.csv	51001 data	32 features(columns)

Each row in the dataset corresponds to a property that was inspected and given a hazard score ("Hazard").
On noticing the train data hazard score as a **continuous number** that represents the condition of the property as determined by the inspection. That's why we felt that this problem fall under numeric prediction.

Some inspection hazards are major and contribute more to the total score, while some are minor and contribute less. The total score for a property is the sum of the individual hazards.
The goal of this project is to forecast the hazard score based on anonymized variables which are available before an inspection is ordered [2].

train.csv - the training set, contains the Hazard and anonymized predictor variables (Features of each property)
test.csv - the test set, contains only the anonymized predictor variables (Features of each property)

On looking into the features (anonymized predictor variables) - It has 32 columns T1_V1 to T1_V17 and T2_V1 to T2_V15.It is a mix of Numerical and Categorical Columns(Nominal Attributes).

Categorical Columns-Nominal Attributes-(All data are in the form of Alphabets) - ['T1_V4', 'T1_V5', 'T1_V6', 'T1_V7', 'T1_V8', 'T1_V9', 'T1_V11',T1_V12', 'T1_V15', 'T1_V16', 'T1_V17', 'T2_V3', 'T2_V5', 'T2_V11', 'T2_V12','T2_V13']

Numerical Columns- ['T1_V1', 'T1_V2', 'T1_V3', 'T1_V10', 'T1_V13', 'T1_V14', 'T2_V1', 'T2_V2', 'T2_V4', 'T2_V6','T2_V7', 'T2_V8', 'T2_V9', 'T2_V10','T2_V14','T2_V15']

CLEANING STEPS:

As a first step to this project we converted all the **categorical columns** (Nominal/Ordinal Attributes) **to numeric.**We used Label Encoder to convert all the Categorical data to numerical value.

ALGORITHM AND TOOLS USED:

Language and Tools: Python-Numpy, Scipy, pandas,XGBoost **scikit learn**-for Machine Learning/DM algorithms. We used Spyder and PyCharm IDE for this project.

Algorithm:
As we found this is a numeric prediction, we felt to start with Decision trees for numeric prediction-Regression Trees
Decision trees, where the target variable can take continuous values (typically real numbers) are called **regression trees** [3].

Using scikit learn we used DecisonTreeRegressor(max_depth=15 - randomly chosen) to implement the Regression tree from the training dataset and predict the Hazard Score for the Test Data and submit it to Kaggle.

But it showed very less results and then we focussed on the randomforest regressor() as mentioned in the contest to use random forest to cross the random forest benchmark.

RANDOM FOREST REGRESSOR:
Scikit Learn -sklearn.ensemble.RandomForestRegressor [4]
A Random forest is good option for any prediction problem. It belongs to a larger class of machine learning algorithms called **ensemble** methods [5].
Each classifier in the ensemble is a decision tree classifier and is generated using a random selection of attributes at each node to determine the split.

Random forest Algorithm will create a bunch of random decision trees. Random forest aggregates classification or regression trees.

`ensemble.RandomForestRegressor(n_estimators=100, max_depth=15)` in which n_estimators determines the number of trees in the forest with a max_depth of 15.

GRADIENT BOOSTING REGRESSOR:

Scikit Learn - sklearn.ensemble.GradientBoostingRegressor[6].

The algorithm for Boosting Trees evolved from the application of boosting methods to regression trees. The general idea is to compute a sequence of (very) simple trees, where each successive tree is built for the prediction residuals of the preceding tree.

```
gb = ensemble.GradientBoostingRegressor(n_estimators=150, max_depth=6)
```

NEURAL NETWORK:

sklearn.neural_network import BernoulliRBM[7]

With initial weight assumption and learning rate 0.1 we build a simple feedforward neural Network.

```
log = linear_model.LogisticRegression()
NN_BRBM_model_1 = BernoulliRBM(n_components=2, learning_rate = 0.1)
cls1 = Pipeline(steps=[('rbm', NN_BRBM_model_1), ('logistic',
log)]).fit(train_data1,y)
```

XGBOOST:

`xgboost as xgb[8]`

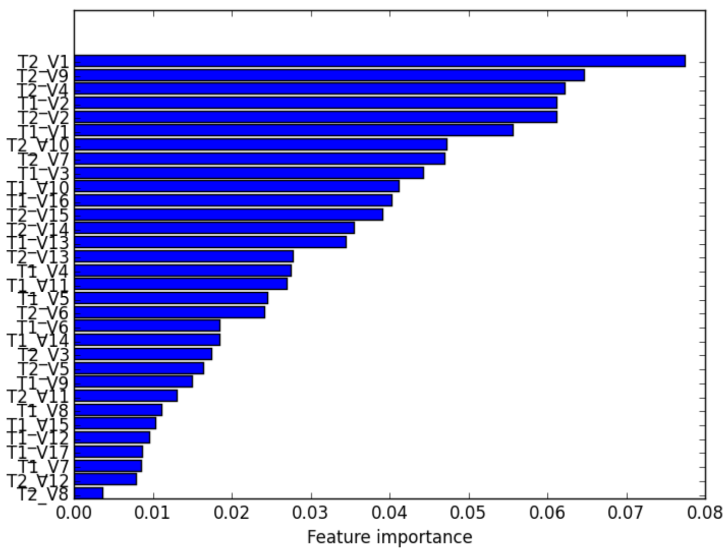
We used XGB to improve our Gini score our in our leaderboard.

Extreme Gradient Boosting—Tree ensemble method.

preds2 = modelXGB.predict(xgtest,ntree_limit=modelXGB.best_iteration)

We iterate until we get a best lesser RMSE Score.

Feature selection based on Relevance Importance Calculated from Random Forest.We choose the top 25 features for selection.

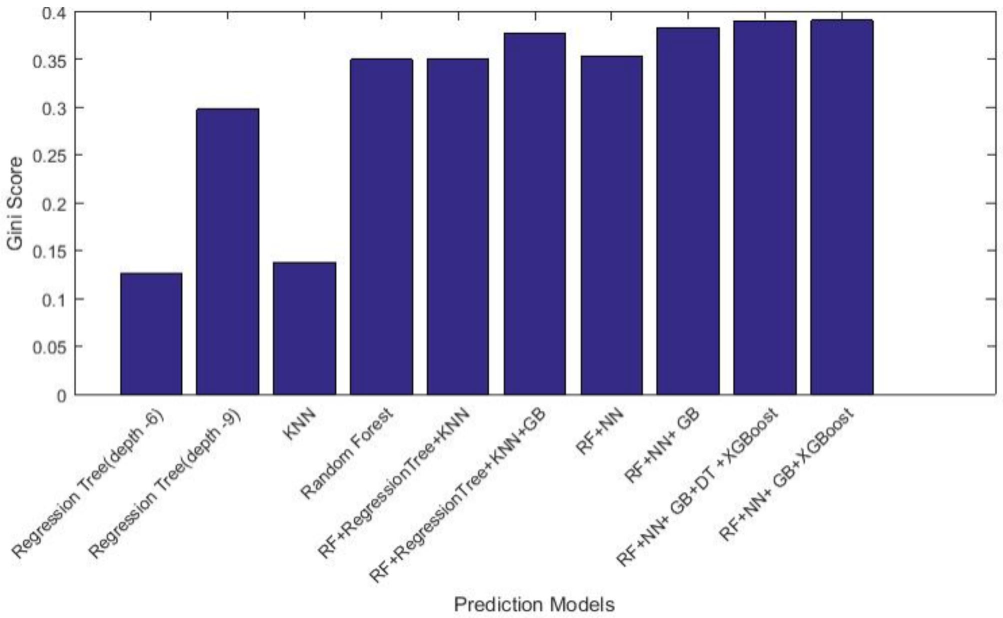


<u>RESULTS ON TRAINING DATA:</u>

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

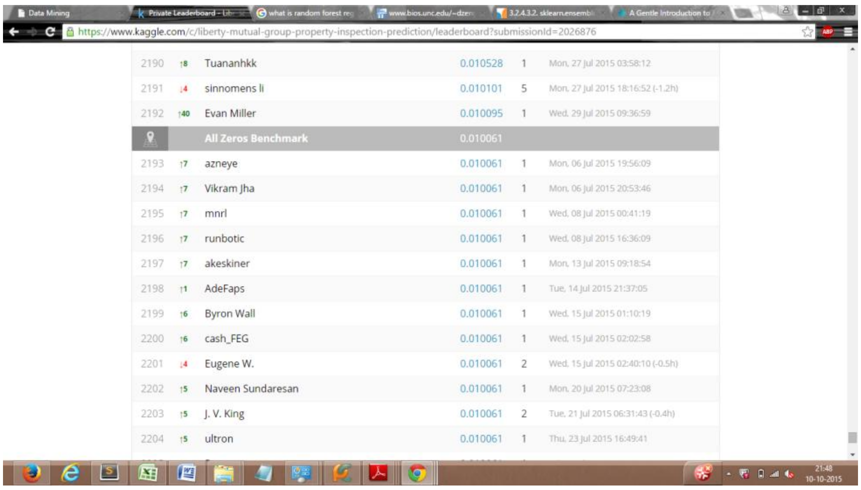
We used RMSE Error metric to evaluate our training model and to get lesser RMSE value(Final Ensemble model gave us RMSE of 0.7342

And Based on our Leader board Score (Gini Score)we preferred the model.



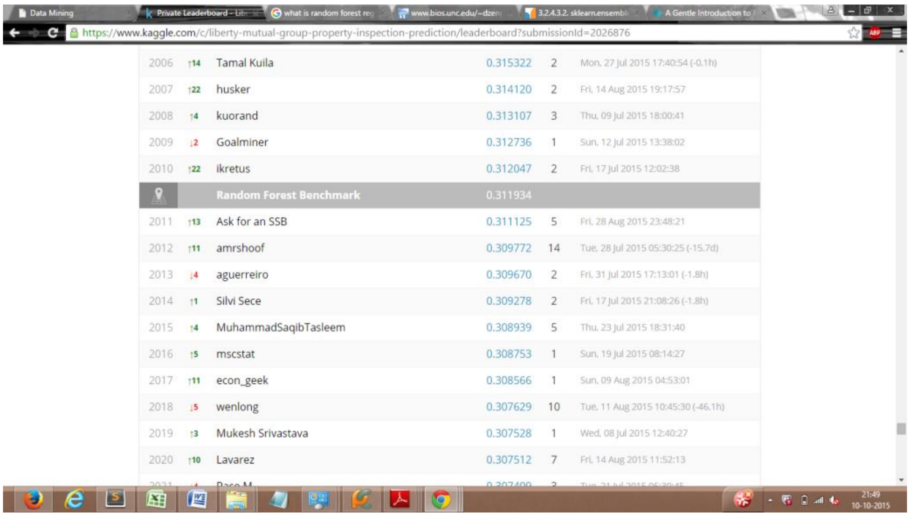
<u>SCREENSHOTS:</u>

Zero Bench Mark:



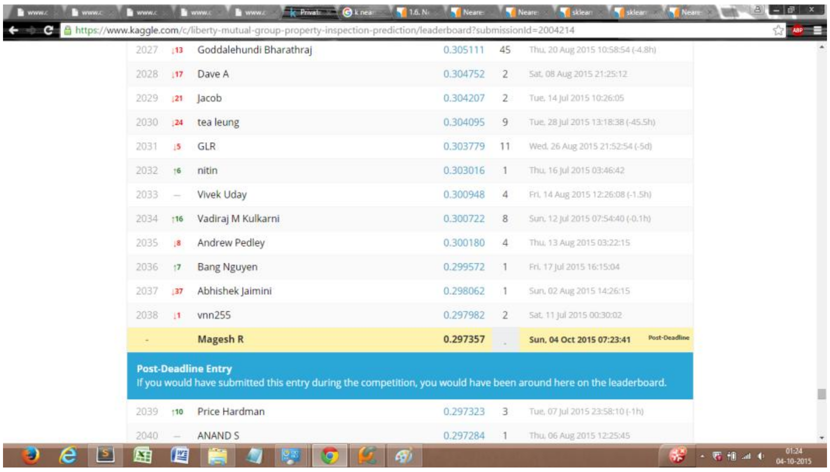
2190	↑8	Tuananhkk	0.010528	1	Mon, 27 Jul 2015 03:58:12
2191	↓4	sinnomens li	0.010101	5	Mon, 27 Jul 2015 18:16:52 (-1.2h)
2192	↑40	Evan Miller	0.010095	1	Wed, 29 Jul 2015 09:36:59
		All Zeros Benchmark	0.010061		
2193	↑7	azneye	0.010061	1	Mon, 06 Jul 2015 19:56:09
2194	↑7	Vikram Jha	0.010061	1	Mon, 06 Jul 2015 20:53:46
2195	↑7	mnrl	0.010061	1	Wed, 08 Jul 2015 00:41:19
2196	↑7	runbotic	0.010061	1	Wed, 08 Jul 2015 16:36:09
2197	↑7	akeskiner	0.010061	1	Mon, 13 Jul 2015 09:18:54
2198	↑1	AdeFaps	0.010061	1	Tue, 14 Jul 2015 21:37:05
2199	↑6	Byron Wall	0.010061	1	Wed, 15 Jul 2015 01:10:19
2200	↑6	cash_FEG	0.010061	1	Wed, 15 Jul 2015 02:02:58
2201	↓4	Eugene W.	0.010061	2	Wed, 15 Jul 2015 02:40:10 (-0.5h)
2202	↑5	Naveen Sundaresan	0.010061	1	Mon, 20 Jul 2015 07:23:08
2203	↑5	J. V. King	0.010061	2	Tue, 21 Jul 2015 06:31:43 (-0.4h)
2204	↑5	ultron	0.010061	1	Thu, 23 Jul 2015 16:49:41

Random Forest Bench Mark:


https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction/leaderboard?submissionId=2026876

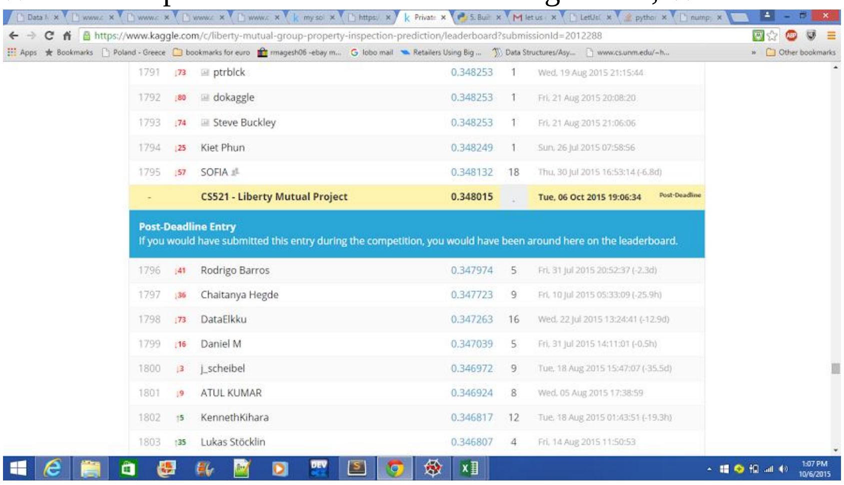
2006	↑14	Tamal Kuila	0.315322	2	Mon, 27 Jul 2015 17:40:54 (-0.1h)
2007	↑22	husker	0.314120	2	Fri, 14 Aug 2015 19:17:57
2008	↑4	kuorand	0.313107	3	Thu, 09 Jul 2015 18:00:41
2009	↓2	Goalminer	0.312736	1	Sun, 12 Jul 2015 13:38:02
2010	↑22	ikretus	0.312047	2	Fri, 17 Jul 2015 12:02:38
		Random Forest Benchmark	0.311934		
2011	↑13	Ask for an SSB	0.311125	5	Fri, 28 Aug 2015 23:48:21
2012	↑11	amrshoof	0.309772	14	Tue, 28 Jul 2015 05:30:25 (-15.7d)
2013	↓4	aguerreiro	0.309670	2	Fri, 31 Jul 2015 17:13:01 (-1.8h)
2014	↑1	Silvi Sece	0.309278	2	Fri, 17 Jul 2015 21:08:26 (-1.8h)
2015	↑4	MuhammadSaqibTasleem	0.308939	5	Thu, 23 Jul 2015 18:31:40
2016	↑5	mscstat	0.308753	1	Sun, 19 Jul 2015 08:14:27
2017	↑11	econ_geek	0.308566	1	Sun, 09 Aug 2015 04:53:01
2018	↓5	wenlong	0.307629	10	Tue, 11 Aug 2015 10:45:30 (-46.1h)
2019	↑3	Mukesh Srivastava	0.307528	1	Wed, 08 Jul 2015 12:40:27
2020	↑10	Lavarez	0.307512	7	Fri, 14 Aug 2015 11:52:13

When Implemented decision Tree regression, we crossed Zero Bench Mark but not Random forest Bench Mark:


https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction/leaderboard?submissionId=2004214

2027	↓13	Goddalehundi Bharathraj	0.305111	45	Thu, 20 Aug 2015 10:58:54 (-4.8h)
2028	↓17	Dave A	0.304752	2	Sat, 08 Aug 2015 21:25:12
2029	↓21	Jacob	0.304207	2	Tue, 14 Jul 2015 10:26:05
2030	↓24	tea leung	0.304095	9	Tue, 28 Jul 2015 13:18:38 (-45.5h)
2031	↓5	GLR	0.303779	11	Wed, 26 Aug 2015 21:52:54 (-5d)
2032	↑6	nitin	0.303016	1	Thu, 16 Jul 2015 03:46:42
2033	—	Vivek Uday	0.300948	4	Fri, 14 Aug 2015 12:26:08 (-1.5h)
2034	↑16	Vadiraj M Kulkarni	0.300722	8	Sun, 12 Jul 2015 07:54:40 (-0.1h)
2035	↓8	Andrew Pedley	0.300180	4	Thu, 13 Aug 2015 03:22:15
2036	↑7	Bang Nguyen	0.299572	1	Fri, 17 Jul 2015 16:15:04
2037	↓37	Abhishek Jaimini	0.298062	1	Sun, 02 Aug 2015 14:26:15
2038	↓1	vnn255	0.297982	2	Sat, 11 Jul 2015 00:30:02
-		**Magesh R**	**0.297357**	.	**Sun, 04 Oct 2015 07:23:41** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
2039	↑10	Price Hardman	0.297323	3	Tue, 07 Jul 2015 23:58:10 (-1h)
2040	—	ANAND S	0.297284	1	Thu, 06 Aug 2015 12:25:45

When we implemented Random Forest Regression, We crossed the Random Forest Regressor.


https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction/leaderboard?submissionId=2012288

1791	↓73	ptrblck	0.348253	1	Wed, 19 Aug 2015 21:15:44
1792	↓80	dokaggle	0.348253	1	Fri, 21 Aug 2015 20:08:20
1793	↓74	Steve Buckley	0.348253	1	Fri, 21 Aug 2015 21:06:06
1794	↓25	Kiet Phun	0.348249	1	Sun, 26 Jul 2015 07:58:56
1795	↓57	SOFIA	0.348132	18	Thu, 30 Jul 2015 16:53:14 (-6.8d)
-		**CS521 - Liberty Mutual Project**	**0.348015**	.	**Tue, 06 Oct 2015 19:06:34** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
1796	↓41	Rodrigo Barros	0.347974	5	Fri, 31 Jul 2015 20:52:37 (-2.3d)
1797	↓36	Chaitanya Hegde	0.347723	9	Fri, 10 Jul 2015 05:33:09 (-25.9h)
1798	↓73	DataElkku	0.347263	16	Wed, 22 Jul 2015 13:24:41 (-12.9d)
1799	↓16	Daniel M	0.347039	5	Fri, 31 Jul 2015 14:11:01 (-0.5h)
1800	↓3	j_scheibel	0.346972	9	Tue, 18 Aug 2015 15:47:07 (-35.5d)
1801	↓9	ATUL KUMAR	0.346924	8	Wed, 05 Aug 2015 17:38:59
1802	↑5	KennethKihara	0.346817	12	Tue, 18 Aug 2015 01:43:51 (-19.3h)
1803	↑35	Lukas Stöcklin	0.346807	4	Fri, 14 Aug 2015 11:50:53

When we tried with our Ensemble model with XGBoost and NN and RF we get higher score

749	↓26	kxx	0.390299	177	Fri, 28 Aug 2015 00:22:30 (-2d)
750	↓22	Grigory Dymov (I want to PZAD)	0.390279	26	Fri, 28 Aug 2015 20:29:15 (-33h)
751	↑23	David Foster	0.390250	29	Thu, 27 Aug 2015 07:20:04 (-11.3d)
752	↓55	anmiko	0.390225	47	Fri, 28 Aug 2015 03:59:44
753	↑99	Michael R	0.390183	27	Fri, 28 Aug 2015 23:02:57
-		**CS521 - Liberty Mutual Project**	**0.390179**	-	**Sat, 21 Nov 2015 23:10:00** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
754	↑37	Alberto Bosko Boschetti	0.390175	24	Wed, 26 Aug 2015 23:47:10 (-13.6d)
755	↓50	Timothy Riley	0.390162	34	Tue, 28 Jul 2015 00:27:11 (-35.1h)
756	↑42	Fizmats	0.390161	22	Fri, 28 Aug 2015 11:58:37 (-10.8d)
757	↑69	BigPlanet	0.390155	15	Mon, 17 Aug 2015 05:46:29 (-16.8h)
758	↑6	GreatDataAnalyst	0.390089	59	Fri, 28 Aug 2015 09:58:00
759	↑111	jdmull	0.390083	75	Tue, 25 Aug 2015 00:06:32 (-0.4h)

NEXT SET OF TASKS TO IMPROVE OUR SCORE:

Try different XGB Model for prediction with different pre-processing data using Label Encoder, Dict Vectorizer.

References:

[1] http://www.cs.stir.ac.uk/courses/CSC9T6/lectures/1%20Data%20Mining/4%20-%20Prediction.pdf
[2] https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction
[3] https://en.wikipedia.org/wiki/Decision_tree_learning
[4] http://scikit learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor
[5] http://blog.yhathq.com/posts/random-forests-in-python.html
[6] https://www.statsoft.com/Textbook/Boosting-Trees-Regression-Classification/button/1
[7] http://scikit-neuralnetwork.readthedocs.org/en/latest/guide_beginners.html

Appendix:

Code:

```
import pandas as pd
#import pydot#
import numpy as np
#import graphviz
import math
from sklearn.externals.six import StringIO
from sklearn import ensemble
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
from os import system
from sklearn import preprocessing
```

```python
from sklearn import tree
from sklearn import svm
from sklearn import neighbors, datasets
from sklearn.neighbors import KNeighborsRegressor
from sklearn.feature_extraction import DictVectorizer as DV
from sklearn.cross_validation import KFold
from sklearn.decomposition import PCA
from sklearn import linear_model
from sklearn.neural_network import BernoulliRBM
from sklearn import cross_validation
from sklearn.pipeline import Pipeline
from sklearn import linear_model
import xgboost as xgb
from sklearn.feature_extraction import DictVectorizer
from datetime import datetime

import matplotlib.pyplot as plt



#Reading the training and Testing Data
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

ids = test['Id']
y = train['Hazard']
train = train.drop(['Hazard', 'Id'], axis=1)
test = test.drop(['Id'], axis=1)

#get the categorical columns
fact_cols = ['T1_V4', 'T1_V5', 'T1_V6', 'T1_V7', 'T1_V8', 'T1_V9',
'T1_V11','T1_V12', 'T1_V15','T1_V16', 'T1_V17', 'T2_V3', 'T2_V5', 'T2_V11',
'T2_V12','T2_V13']

#time evalution
start_time = datetime.now()


#Preprocessing of the data using Label Encoder
lbl = preprocessing.LabelEncoder()
for column in fact_cols:
    train[column+'_new'] = lbl.fit_transform(train[column])
    test[column+'_new'] = lbl.fit_transform(test[column])
train_data = train.drop(fact_cols,axis=1)
test_data = test.drop(fact_cols,axis=1)
train_data = train_data.astype(float)
test_data = test_data.astype(float)

train_data.drop('T2_V10', axis=1, inplace=True)
train_data.drop('T2_V7', axis=1, inplace=True)
train_data.drop('T1_V13', axis=1, inplace=True)
train_data.drop('T1_V10', axis=1, inplace=True)

test_data.drop('T2_V10', axis=1, inplace=True)
test_data.drop('T2_V7', axis=1, inplace=True)
test_data.drop('T1_V13', axis=1, inplace=True)
test_data.drop('T1_V10', axis=1, inplace=True)

############ PCA REDUCTION ##########
#pca=PCA(n_components=2)
#pca.fit(train_data)
```

```
#pca.fit(test_data)
#plt.plot(train_data,'ro')
#plt.show()

#put the numerical as matrix

train_data1 = np.array(train_data.as_matrix(columns = None),
dtype=object).astype(np.int)
test_data1 = np.array(test_data.as_matrix(columns=None),
dtype=object).astype(np.int)




########### RANDOM FOREST REGRESSOR ##########
n_neighbors=8

rf = ensemble.RandomForestRegressor(n_estimators=200, max_depth=15)
rf.fit(train_data1, y)
pred1 = rf.predict(test_data1)

########### DECISION TREE REGRESSOR ##########
#dt =tree.DecisionTreeRegressor(max_depth=15)
#dt.fit(train_data1, y)

########### K NEIGHBOUR REGRESSOR ##########
#pred2 = dt.predict(test_data1)
#knn =KNeighborsRegressor(n_neighbors=9)
#knn.fit(train_data1, y)
#pred3 = knn.predict(test_data1)

########### GRADIENT BOOSTING REGRESSOR ##########
gb = ensemble.GradientBoostingRegressor(n_estimators=150, max_depth=6)
gb.fit(train_data1, y)
pred4 = gb.predict(test_data1)

########### SVM MODEL ##########
#svm1 = svm.SVR()
#svm1.fit(train_data1, y)
#pred5 = svm1.predict(test_data1)

##########XGBOOST##########################
def xgboost_pred(train,labels,test):
   params = {}
   params["objective"] = "reg:linear"
   params["eta"] = 0.005
   params["min_child_weight"] = 6
   params["subsample"] = 0.7
   params["colsample_bytree"] = 0.7
   params["scale_pos_weight"] = 1
   params["silent"] = 1
   params["max_depth"] = 9


   listOfParameters = list(params.items())

   offset = 4000

   num_rounds = 10000
   xgtest = xgb.DMatrix(test)
```

```
    #create a train and validation dmatrices
    xgtrain = xgb.DMatrix(train[offset:,:], label=labels[offset:])
    xgval = xgb.DMatrix(train[:offset,:], label=labels[:offset])

    #predicition
    evallist = [(xgtrain, 'train'),(xgval, 'val')]
    modelXGB = xgb.train(listOfParameters, xgtrain, num_rounds, evallist,
early_stopping_rounds=10)
    preds1 = modelXGB.predict(xgtest,ntree_limit=modelXGB.best_iteration)


    #reverse train and labels
    train = train[::-1,:]
    labels = np.log(labels[::-1])

    xgtrain = xgb.DMatrix(train[offset:,:], label=labels[offset:])
    xgval = xgb.DMatrix(train[:offset,:], label=labels[:offset])

    evallist = [(xgtrain, 'train'),(xgval, 'val')]
    modelXGB = xgb.train(listOfParameters, xgtrain, num_rounds, evallist,
early_stopping_rounds=120)
    preds2 = modelXGB.predict(xgtest,ntree_limit=modelXGB.best_iteration)


    #combine predictions
    preds = preds1*1.4 + preds2*8.6
    return preds
########### XGB MODEL ##########
pred7=xgboost_pred(train_data1,y,test_data1)


########### NEURAL NETWORK MODEL ##########
log = linear_model.LogisticRegression()
NN_BRBM_model_1 = BernoulliRBM(n_components=3, learning_rate = 0.1)
cls1 = Pipeline(steps=[('rbm', NN_BRBM_model_1), ('logistic',
log)]).fit(train_data1,y)


########### CROSS VALIDATING NN MODEL ##########

KF_NN1 = cross_validation.KFold(len(train_data1), n_folds=10, shuffle=True,
random_state=4)
Score_NN1 = cross_validation.cross_val_score(cls1, train_data1, y,
cv=KF_NN1, n_jobs=1)

print "Neural Network Accuracy : "+str(Score_NN1.mean())

pred6=cls1.predict(test_data1)

######## Comination of Predicitions #############
pred=(pred1*0.05)+(pred4*0.15)+(pred6*0.5)+(pred7*0.3)

preds = pd.DataFrame({"Id": ids, "Hazard": pred})

preds = preds[['Id', 'Hazard']]

preds.to_csv('result_random_with_time.csv', index=False)

end_time = datetime.now()
time_taken = (end_time - start_time)
print time_taken
```