Predictive Modeling of Property Hazards Prior to Inspection

Prepared for:

Liberty Mutual Insurance
175 Berkeley Street
Boston, Massachusetts 02116

Prepared by:

Justin Geiman
Fire & Risk Alliance
7361 Calhoun Place, Suite 690
Rockville, MD 20855

Capstone Project Report submitted in partial fulfillment of the requirements of SlideRule's Data Science Intensive

November 28, 2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

This report documents a Capstone Project for SlideRule's Data Science Intensive course. The purpose of the Capstone is to apply the knowledge gained throughout the course to a real-world data science problem.

As with many industries, the insurance industry is increasingly turning to data science to improve their business processes. For example, Liberty Mutual Insurance is applying predictive modeling in its Actuarial, Product, Claims, Marketing, Distribution, Human Resources, and Finance departments. To further this work, Liberty Mutual Insurance sponsored a Kaggle competition on predictive modeling of property hazards[1]. This competition involved predicting a count of hazards or pre-existing damages using property information. Liberty Mutual wants to use these predictions to identify high-risk homes that require additional examination to confirm their insurability. These inspections assess attributes of the property such as the condition of the foundation, roof, windows and siding.

Liberty Mutual developed the data set used in this project from properties that had been inspected and were given a hazard score. The total hazard score for a property is the sum of the individual hazards, and some hazards identified in the inspection contribute more to the hazard score than others. The goal of the competition was to forecast the hazard score from variables that are available before the inspection.

This report thoroughly documents one approach to the predictive modeling of property hazards from Liberty Mutual's Kaggle competition. First, an exploration of the data set is presented. The technical approach is then provided, followed by a presentation of the results and discussion. Finally, conclusions and recommendations are presented based on the lessons learned throughout this project.

[1] https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction

2.0 DATA EXPLORATION

Liberty Mutual provided the data set used for this project as part of the Kaggle competition. The data set was split into a training data set and a testing data set. Each of these data sets was provided as comma-separated value (CSV) files. The training data contained 51,000 property records, that each contained a hazard score and 32 anonymous predictor variables. The test data contained 51,000 property records, that each contained only the 32 anonymous predictor variables. The sections that follow provide further exploration of the data set.

2.1 Data Cleaning & Wrangling

Most data sets some amount of cleaning and wrangling to transform the data into a usable form. The data set provided by Liberty Mutual for the Kaggle competition was very clean. There were no missing or obviously incorrect values. The only transformations that were required to prepare the data set were encoding the categorical variables. Two different encoding strategies were evaluated:

- Map each unique categorical value to an integer value
- Create dummy/indicator variables for each categorical variable

The first approach, which was used for the majority of the models created, replaced the categorical values such as ‘A’, ‘B’, ‘C’, ‘D’ with integer values 1, 2, 3, 4. Variables that only contained the unique values ‘Y’, ‘N’ were encoded as 1 and 0 (zero), respectively. The second approach was tested for several modeling attempts but did not provide significantly improve the results and was abandoned for subsequent models. Using the second approach, a column with three unique variables would be transformed into three separate columns with each column containing either a 1 or 0 (zero) depending on the original value.

2.2 Hazard Score

The goal of this project is to predict a hazard score from information available prior to an inspection. Properties with a high hazard score pose a significant risk for a major loss. Accurate determination of the hazard associated with a property is a necessary part of the underwriting process used by Liberty Mutual.

Hazard scores in the training data set ranged from 1 to 69. The median hazard score was 3, and the mean was 4. The mode of the data set was 1; 18981 properties in the data set had this hazard score. This means almost 40% of the properties in the data set had the lowest possible hazard score. Given the range of values and the mean/median/mode the data set seems highly skewed. A histogram of the distribution of hazard scores is plotted in Figure 1, along with a fit to an exponential distribution. Figure 2 shows the cumulative distribution function (CDF) of hazard scores, as well as the CDF for the exponential distribution fit the hazard scores. Based on Figure 1 and Figure 2, the hazard scores in the training data set are indeed skewed, and in fact, appear to fit an exponential distribution well. The exponential distribution is often used in reliability applications to model data with a constant failure rate.

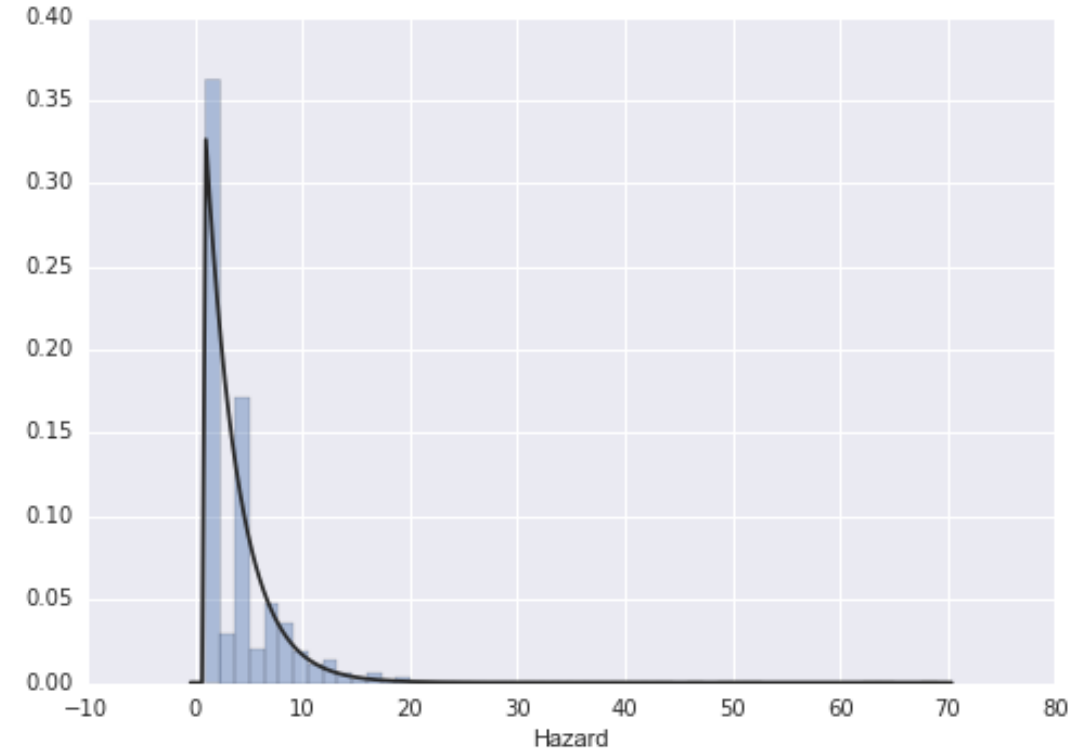


Figure 1. Histogram of Hazard Scores. Solid line shows exponential distribution fit.

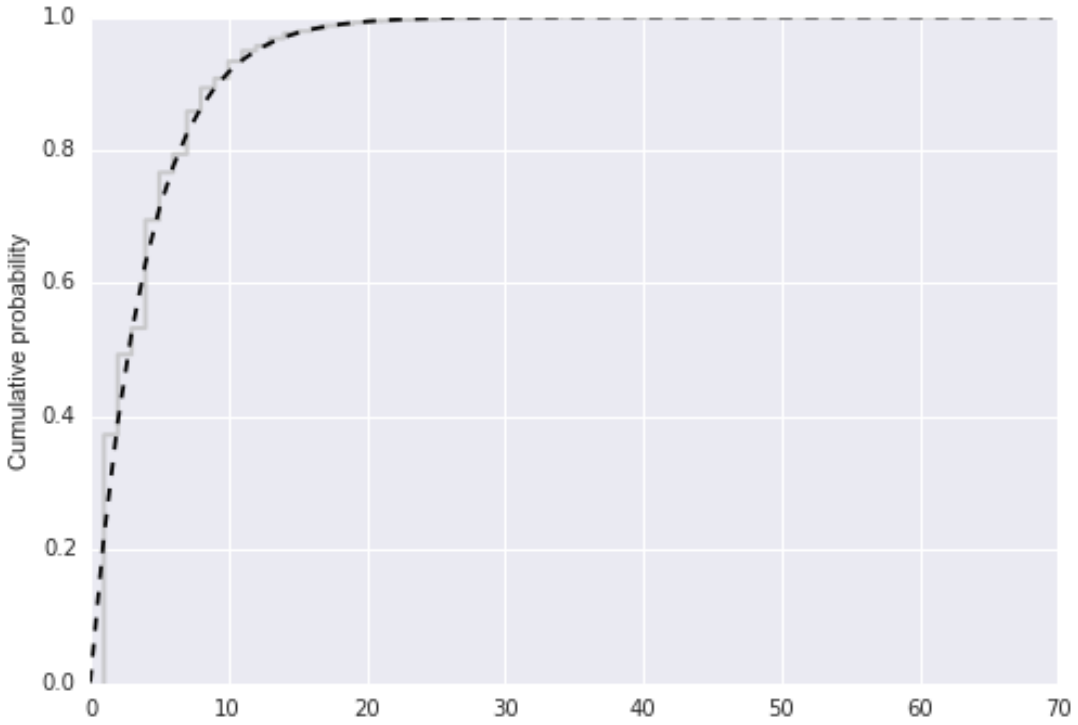


Figure 2. Cumulative Distribution Function (CDF) of Hazard Scores. Dashed line shows exponential distribution fit.

2.3 Features

The training and testing data sets contain 32 anonymous feature variables. Sixteen of these features were numerical variables, and 16 were categorical variables. The numerical values were all integer values. Table 1 summarizes information on the numerical features in the data set, and Table 2 summarizes information on the categorical variables.

Table 1. Numerical Variable Information

Feature	Numeric Type	Minimum	Maximum	Mean	Std. Dev.
T1_V1	Integer	1	19	9.7	5.2
T1_V2	Integer	1	24	12.8	6.3
T1_V3	Integer	1	9	3.2	1.7
T1_V10	Integer	2	12	7.0	3.6
T1_V13	Integer	5	20	14.0	4.6
T1_V14	Integer	0	4	1.6	0.9
T2_V1	Integer	1	100	57.6	23.5
T2_V2	Integer	1	39	12.4	4.8
T2_V4	Integer	1	22	10.3	4.9
T2_V6	Integer	1	7	1.9	0.8
T2_V7	Integer	22	40	33.5	5.8
T2_V8	Integer	1	3	1.0	0.2
T2_V9	Integer	1	25	12.5	7.3
T2_V10	Integer	1	7	4.5	1.9
T2_V14	Integer	1	7	2.5	1.3
T2_V15	Integer	1	12	3.5	3.1

Table 2. Categorical Variable Information

Feature	Unique Values Count	Unique Values
T1_V4	8	B, C, E, G, H, N, S, W
T1_V5	10	A, B, C, D, E, H, I, J, K, L
T1_V6	2	N, Y
T1_V7	4	A, B, C, D
T1_V8	4	A, B, C, D
T1_V9	6	B, C, D, E, F, G
T1_V11	12	A, B, D, E, F, H, I, J, K, L, M, N
T1_V12	4	A, B, C, D
T1_V15	8	A, C, D, F, H, N, S, W
T1_V16	18	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R
T1_V17	2	N, Y
T2_V3	2	N, Y
T2_V5	6	A, B, C, D, E, F
T2_V11	2	N, Y
T2_V12	2	N, Y
T2_V13	5	A, B, C, D, E

The Python Seaborn[2] visualization library was used to generate a plot for each feature variable that contained bivariate (with hazard score) and univariate graphs. These plots are referred to as JointPlots in Seaborn. Figure 3 is a joint plot for feature variable T1_V3 and Figure 4 is a joint plot for feature variable T2_V1. Appendix A contains joint plots of all variables with hazard score. Overall, these plots did not provide any obvious patterns that could be exploited in developing predictive models.

[2] http://stanford.edu/~mwaskom/software/seaborn/index.html

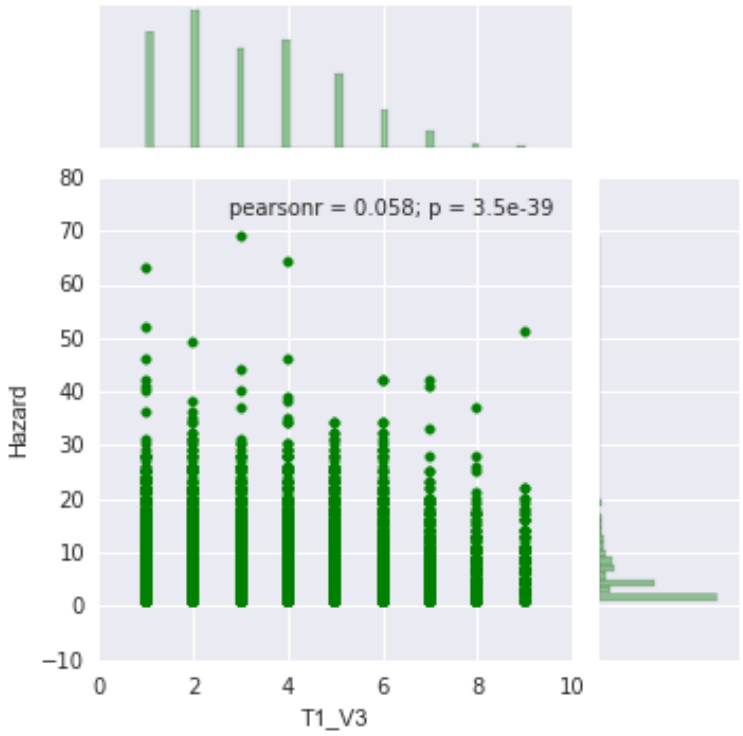


Figure 3. Joint Plot of Feature T1_V3 and Hazard Score

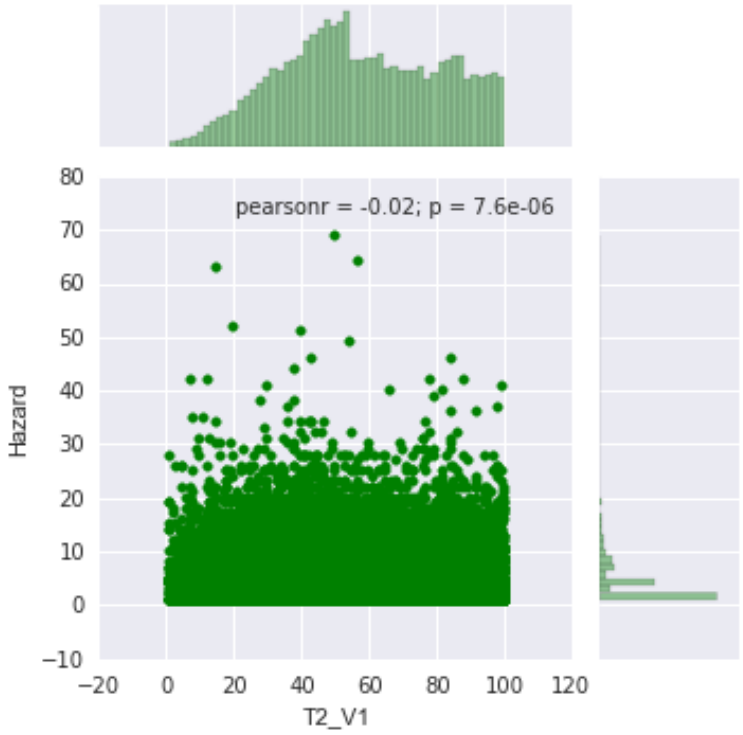


Figure 4. Joint Plot of Feature T2_V1 and Hazard Score

The correlation of feature variables was also explored. Table 3 shows the correlation between variables in the data set. The table is color-coded to provide a visual indication of the strength of the correlation between variables. Variables with a perfect positive correlation (+1.0) are shown in green, and variables with a perfect negative correlation (-1.0) are shown in red. Correlation values between these extremes are shaded accordingly. For example, variables with no correlation (correlation = 0) are shaded a yellow color.

The first row & column of Table 3 show the correlation of the feature variables with the hazard score. There is no strong correlation of any feature variable with the hazard score (i.e. all correlations <= 0.1). Note that most features show no correlation (correlation = 0) or a weak positive correlation (correlation <= +0.1) with hazard score. The exceptions to this observation are features T1_V14, T1_V15, and T2_V9, which have weak negative correlations (correlation ~ -0.1) with the hazard score.

Highly correlated variables, for example two variables that were simple multiples of one another, can cause problems with some machine learning algorithms, particularly linear models. Due to this problem, sometimes referred to as multicollinearity, people often remove highly correlated features from a data set. However, since there was a relative small number of variables, all of which were anonymous, and linear models were not the focus of the modeling approach (discussed in detail in Section 3.0), no features variables were removed.

Table 3. Correlation of Feature Variables

	Hazard	T1_V1	T1_V2	T1_V3	T1_V4	T1_V5	T1_V6	T1_V7	T1_V8	T1_V9	T1_V10	T1_V11	T1_V12	T1_V13	T1_V14	T1_V15	T1_V16	T1_V17	T2_V1	T2_V2	T2_V3	T2_V4	T2_V5	T2_V6	T2_V7	T2_V8	T2_V9	T2_V10	T2_V11	T2_V12	T2_V13	T2_V14	T2_V15
Hazard	1.0	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	-0.1	-0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.1
T1_V1	0.1	1.0	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	-0.1	-0.2	0.1	0.0	-0.1	0.1	0.0	-0.1	0.0	0.0	0.0	0.0	-0.2	0.0	-0.2	0.0	-0.2	0.0	0.0
T1_V2	0.1	0.0	1.0	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	-0.1	-0.1	0.1	0.0	0.1	0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
T1_V3	0.1	0.0	0.1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
T1_V4	0.1	0.2	0.0	0.0	1.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	-0.2	-0.2	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	-0.1	0.0	-0.1	0.0	-0.1	0.0	0.0
T1_V5	0.1	0.1	0.1	0.0	0.1	1.0	0.0	0.0	0.0	0.7	0.0	0.4	0.0	0.0	-0.1	0.0	0.7	0.0	0.1	0.1	0.0	0.1	-0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
T1_V6	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V8	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V9	0.1	0.1	0.1	0.0	0.0	0.7	0.0	0.0	0.0	1.0	0.0	0.4	0.0	0.0	0.0	0.0	0.7	0.0	0.0	-0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1
T1_V10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V11	0.1	0.0	0.1	0.0	0.1	0.4	0.0	0.0	0.0	0.4	0.0	1.0	0.0	0.0	-0.1	0.0	0.4	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T1_V14	-0.1	-0.1	-0.1	-0.1	-0.2	-0.1	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	1.0	0.1	0.0	-0.1	-0.1	-0.1	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.1	-0.1	0.1	0.0	0.0
T1_V15	-0.1	-0.2	-0.1	0.1	-0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
T1_V16	0.1	0.1	0.1	0.0	0.1	0.7	0.0	0.0	0.0	0.7	0.0	0.4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1
T1_V17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	1.0	0.3	-0.1	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.0	0.2	1.0	0.2	0.0	0.0
T2_V1	0.0	-0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.3	1.0	0.0	0.0	0.3	-0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.3	0.2	-0.1	0.2
T2_V2	0.0	0.1	0.2	0.0	0.1	0.1	0.0	0.0	0.0	-0.1	0.0	0.2	0.0	0.0	-0.1	0.1	0.0	-0.1	0.0	1.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	-0.1	0.0	0.1	0.0
T2_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T2_V4	0.0	-0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.3	0.0	0.0	1.0	-0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.1	-0.1	0.5
T2_V5	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.1	-0.1	0.0	0.0	-0.1	1.0	0.0	0.0	0.0	-0.1	0.0	0.0	0.1	0.0	0.0	-0.3
T2_V6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	1.0	0.0	0.3	0.0	0.0	0.0	0.1	0.0	0.6	0.1
T2_V7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T2_V8	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.3	0.0	1.0	0.0	0.0	0.1	0.2	0.1	0.2	0.0
T2_V9	-0.1	-0.2	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	-0.1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.1	0.0	0.1
T2_V10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
T2_V11	0.0	-0.2	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.2	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	1.0	0.2	0.7	0.0	0.1
T2_V12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	-0.1	0.0	0.1	1.0	0.3	-0.1	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.0	0.2	1.0	0.2	0.0	0.0
T2_V13	0.0	-0.2	0.0	0.0	-0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.2	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.7	0.2	1.0	-0.1	0.0
T2_V14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	0.1	0.0	-0.1	0.0	0.6	0.0	0.2	0.0	0.0	0.0	0.0	-0.1	1.0	0.0
T2_V15	0.1	0.0	0.2	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.0	0.5	-0.3	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	1.0

3.0 APPROACH

Experiments were conducted with multiple different modeling approaches in an attempt to improve the performance of the resulting predictive model. The Python programming language was used to implement all predictive models generated for this project. The Python machine learning library scikit-learn was used to implement most of the machine learning algorithms used in this project. This section discusses the various approaches used in modeling property hazard scores.

3.1 Feature Engineering

Feature engineering is typically an extensive part of developing well-performing predictive models for data science problems. One of the difficulties of this particular data set, was that all the feature variables were anonymous. It is difficult to generate useful new features from anonymous variables. There is the potential to combine two variables in nonsensical ways. For example, you could end up combining the zip code and the year the home was built to create a variable that had no meaning and could, in fact, make your predictive models worse. Due to the difficulties in combining anonymous variables to generate new features, no feature engineering was done to the original 32 features provided in the data set.

3.2 Ensemble Methods

The baseline modeling approach was to apply common decision-tree based ensemble methods, such as Random Forests, to the regression problem. Ensemble methods combine several base estimators, in various ways, to improve the robustness of the model. Two categories of ensemble methods typically employed include averaging methods and boosting methods.

Averaging methods average the predictions of independent estimators to reduce variance. The baseline averaging method implemented in this project was Random Forest regression. Another decision tree-based averaging method used in this project was the Extra-Trees algorithm. Both algorithms are based on averaging randomized decision trees to reduce variance and prevent over-fitting.

The random forest algorithm creates an ensemble of decision trees, each built using a bootstrap sample from the data set. As a result of averaging the sampled random subsets of the training data, the variance is reduced and a better decision tree model is obtained. The extremely randomized trees (ExtraTrees) algorithm is similar to a random forest, but takes the randomization further. As a result, the ExtraTrees algorithm is often able to obtain decision trees with further reductions in variance compared to RandomForests, at the expense of higher bias. Further details on the scikit-learn implementation of these algorithms is provided in the skikit-learn documentation[3].

The second category of ensemble methods examined for this project was boosting methods. Boosting methods build estimators sequentially to reduce bias. Two implementations of boosting algorithms were examined: Gradient Tree Boosting (scikit-learn[4]) and eXtreme Gradient

[3] http://scikit-learn.org/stable/modules/ensemble.html#forest
[4] http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

Boosting (XGBoost[5]). These boosting methods are robust to outliers and offer remarkable predictive power for many problems. XGBoost is a popular algorithm of choice among winners of a number of Kaggle competitions.

3.3 Stacked Classification & Regression

The second predictive modeling approach used in this project was to stack classification and regression models. The basic process was to bin the hazard scores into classes. A classification model was then fit to predict the hazard class. Regression models were fit for each hazard class. At prediction time, first the hazard class was predicted with the classification model and then the appropriate regression model for that class was used to predict the hazard score. A custom Python class was developed to automate the stacked classification and regression process outlined above, using the same interface used in scikit-learn.

Table 4 shows the three different hazard classification schemes (bins) that were evaluated for the classification portion of the stacked models.

Table 4. Hazard Score Bins

Bin	Label	Hazard Score Ranges		
		Scheme 1	Scheme 2	Scheme 3
1	Low	<= 2	<= 7	<= 2
2	Medium	2 < X <= 7	7 < X < 20	2 < X < 20
3	High	7 < X < 20	>= 20	>= 20
4	Very High	>= 20		

The following classification models were examined for the stacked classification and regression approach:

- Random Forest
- ExtraTrees
- Gradient Boosted Trees
- XGBoost
- Support Vector Machine (SVM)
- Logistic Regression
- K-Nearest Neighbors (KNN)

The following regression models were examined for the stacked classification and regression approach:

- Random Forest
- ExtraTrees
- Gradient Boosted Trees
- XGBoost

[5] http://xgboost.readthedocs.org/en/latest/#

3.4 Binary Classification as Regression

Another approach that was evaluated was to treat the regression problem as a binary classification problem. An interview with the winner of this competition mentioned that this was the approach he used. The basic process was to create a series of *K* binary classification problems to determine if the hazard score was greater than X, with X varied over the range of values of the hazard score in the training data set. For example, a classification model would be used to generate a binary classification (1 = True, 0 = False) if hazard score was greater than 1. This process would be repeated for hazard scores greater than 2, 3, 4, etc. up to some maximum hazard score. The sum of all these binary classifications provides an estimate of the hazard score. In essence, this approach creates an ensemble of binary classifications for numerous hazard levels and then combines them with a simple summation. For this project, two binary classification approaches were examined:

- Hazard score > X, for X = 1 to 30 by ones, and X = 35 to 70 by fives
- Hazard score > X, for X = 1 to 70 by intervals of ones.

The following classification models were examined for the binary classification as regression approach:

- Random Forest
- ExtraTrees
- Gradient Boosted Trees
- XGBoost

3.5 Ensembling Output from Multiple Models

The final approach used in this project was to create ensembles from multiple models, using the approaches outlined in the previous sections. It can be shown that for uncorrelated models, averaging the output of several models will improve results. Ensembles were created by combining (averaging) the hazard scores output from various models. Other ensembling strategies were attempted such as taking the minimum or maximum hazard score from the models in the ensemble, but averaging proved to be the best strategy.

4.0 RESULTS

This section presents the results from applying the various approaches outlined in the previous section to this particular data science problem.

4.1 Evaluation Metric

The evaluation metric for models is specified as part of the Kaggle competition. For this competition, the normalized Gini coefficient[6] was used to evaluate predictions. With this metric, only the order of the predicted values is important, not their absolute value. To calculate the Gini index, the predicted values are sorted from largest to smallest. The predictions are then examined to determine what percentage of the observed loss is accounted for a given percentile of the predictions. The area under the curve between the predictions and the straight line of the null model (10% of loss accumulated in 10% of predictions) is the Gini coefficient. The Gini coefficient is normalized by the Gini coefficient of a perfect model, the maximum achievable area under the curve.

The normalized Gini coefficient was calculated for the training data set for each model for reference. The evaluation metric by Kaggle for the test data set upon submission of each model. The normalized Gini coefficient of the test data set, as calculated by Kaggle, was used as the true performance measure of each model.

4.2 Submissions

A total of 49 predictive models were created and submitted to Kaggle as part of this project. Table 5 summarizes the models submitted to Kaggle as part of this project. Note that the competition was already completed when this project commenced, so the calculated normalized Gini coefficient on the test data set is the private leaderboard score on the entire test data set. Also note the rank in the competition was based on submissions from 2236 participants in the competition.

The best performing model was an ensemble of four models, including the best performing Random Forest Regressor, XGBoost Regressor, Stacked XGBoost Classification/Regression models, and XGBoost Binary Classification models. This ensemble achieved a normalized Gini coefficient of 0.375. The winner of the competition achieved a normalized Gini coefficient of 0.397 with an ensemble of XGBoost models using the binary classification approach.

Overall, the models implemented in this competition were relatively simple, with fast training times. The maximum training times of any of the submissions was 980 seconds (16 minutes).

All code developed for this project is hosted on Github:
https://github.com/jgeiman/DataScienceIntensive/Capstone

[6] https://www.kaggle.com/wiki/Gini

Table 5. Model Summary

ID	Description	Model Type	Normalized Gini Test	Kaggle Rank	Normalized Gini Train	Training Time (s)
1	Random Forest Regressor, default parameters	Ensemble	0.252	2082	0.888	4.35
2	Random Forest Regressor, 500 trees, 50 samples/split	Ensemble	0.358	1614	0.615	148.43
3	Random Forest Regressor, 500 trees, 75 samples/split	Ensemble	0.361	1588	0.545	136.94
4	Same as 3 with PCA (24 features)	Ensemble	0.284	2049	0.67	774
5	Same as 3 with PCA (30 features)	Ensemble	0.311	2012	0.675	982
7	Stacked Classifier & Regressor, 100 trees	Stacked	0.234	2092	0.986	30.14
8	Stacked Classifier & Regressor, 500 trees	Stacked	0.246	2085	0.988	154.28
9	Same as 8, changed min sample splits	Stacked	0.246	2085	0.992	158.96
10	Change hazard classes (2)	Stacked	0.315	2006	0.914	173.27
11	Changed hazard classes (3)	Stacked	0.321	1996	0.957	32.7
12	Same as 11, changed min sample split for classifier	Stacked	0.331	1901	0.959	63.25
13	Same as 12, hazard classes (2)	Stacked	0.306	2024	0.948	61.07
14	Same as 12, hazard classes (1)	Stacked	0.243	2092	0.992	64.66
15	Same as 14, using ExtraTrees	Stacked	0.226	2099	0.993	56.44
16	Same as 15, hazard class (2)	Stacked	0.311	2012	0.959	51.39
17	Same as 15, hazard class (1)	Stacked	0.323	1988	0.967	55.82
18	Same as 17, GBT, haz class 3	Stacked	0.332	1897	0.496	113.45
19	Same as 18, hazard class 2	Stacked	0.321	1997	0.583	106.44
20	Same as 18 hazard class 1	Stacked	0.266	2071	0.563	149.23
21	Same as 18 hazard class 1	Stacked	Did not submit	-	-	-
22	Stacked, XGBoost, haz class 3, 100 trees	Stacked	0.335	1878	0.471	6.15
23	Same as 22, varied learning rate for each class	Stacked	0.312	2002	0.476	6.14
24	Same as 22, haz class 2	Stacked	0.297	2041	0.617	6.38
25	Same as 22, haz class 1	Stacked	0.268	2067	0.523	7.76
26	Stacked, SVM & XGBoost regressor, hazard class 1	Stacked	0.121	2138	0.979	768.18

ID	Description	Model Type	Normalized Gini Test	Kaggle Rank	Normalized Gini Train	Training Time (s)
27	Stacked, SVM & XGBoost regressor, hazard class 3	Stacked	0.255	2079	0.939	599.97
28	Stacked, LogisticRegression & XGBoost regressor, hazard class 3	Stacked	Did not submit	-	-	-
29	Stacked, KNN & XGBoost, hazard class 3	Stacked	0.204	2108	-	-
30	Stacked, KNN & XGBoost, hazard class 1	Stacked	0.166	2126	0.986	2
31	Same as 22, with randomness added	Stacked	0.326	1976	0.444	5.59
99	RF Classifier with Binarized Hazard Levels, 500 trees	Binary Classification	0.3	2035	-	980.12
100	Same as 99, changed min split to 4	Binary Classification	0.301	2035	-	964.42
101	Same as 99, Gradient Boosted Trees	Binary Classification	-	-	-	-
102	Same as 99, XGBoost	Binary Classification	0.3	2035	-	285.7
103	Same as 102, binary hazards 2 to 70 by 1	Binary Classification	0.3	2035	0.861	386.64
104	Same as 103, binary objective	Binary Classification	0.27	2059	0.861	385.38
105	Same as 103, binary objective	Binary Classification	0.27	2059	0.861	385.38
106	Same as 102, categorical vars as dummies	Binary Classification	0.3	2037	0.86	652.86
107	Same as 102, with randomness added	Binary Classification	0.319	2000	0.527	55.78
201	XGBoost	Ensemble	0.319	2001	0.695	7.16
202	XGBoost with randomness added	Ensemble	0.358	1612	0.453	1.3
203	Same as 202, 500 trees	Ensemble	0.301	2033	0.657	6.48
	Ensemble - Mean of 3, 22, 107, 202	-	0.375	1399	-	-
	Ensemble - Max of 3, 22, 107, 202	-	0.36	1591	-	-
	Ensemble - Mean of 3, 22, 107	-	0.371	1448	-	-
	Ensemble - Max of 3, 22, 107	-	0.356	1629	-	-
	Ensemble - Min of 3, 22, 107	-	0.326	1974	-	-
	Ensemble - Mean of 3, 18, 22, 107	-	0.368	1495	-	-
	Ensemble - Mean of all submissions	-	0.372	1422	-	-

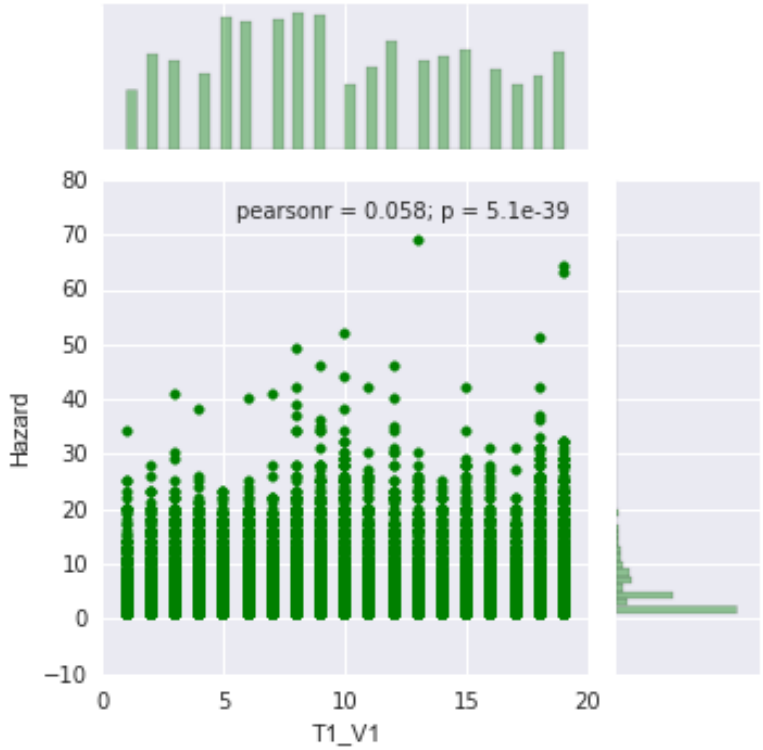
5.0 CONCLUSIONS AND RECOMMENDATIONS

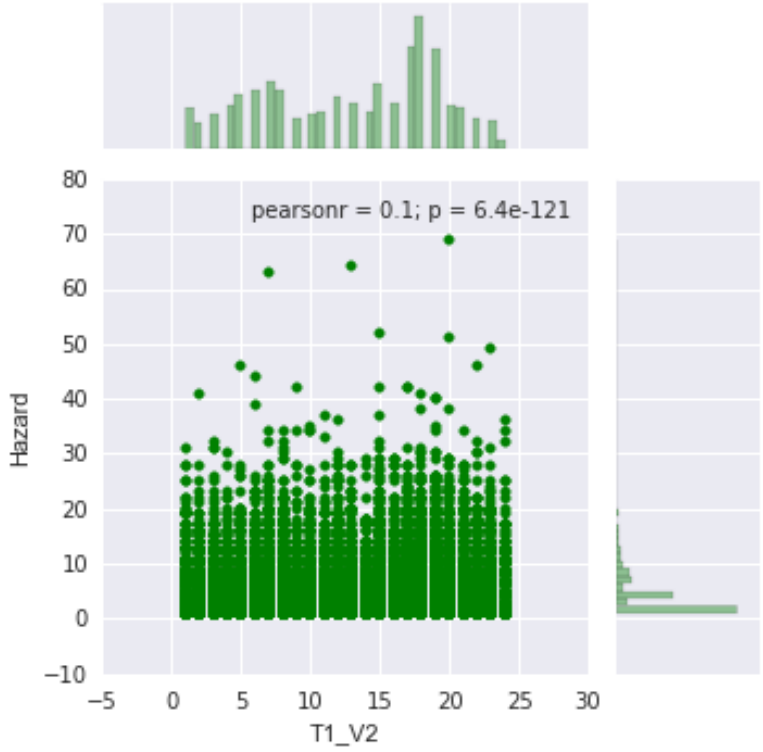
This report documents a Capstone Project for SlideRule's Data Science Intensive course. The data set used for this project was taken from a Kaggle competition on predictive modeling of property hazards sponsored by Liberty Mutual Insurance. The goal of the competition was to forecast the hazard score from variables that are available before the inspection. A total of 49 models were created and submitted to Kaggle for evaluation against the test data set, using a wide variety of machine learning approaches. The best performing model created for this project achieved a normalized Gini coefficient of 0.375.

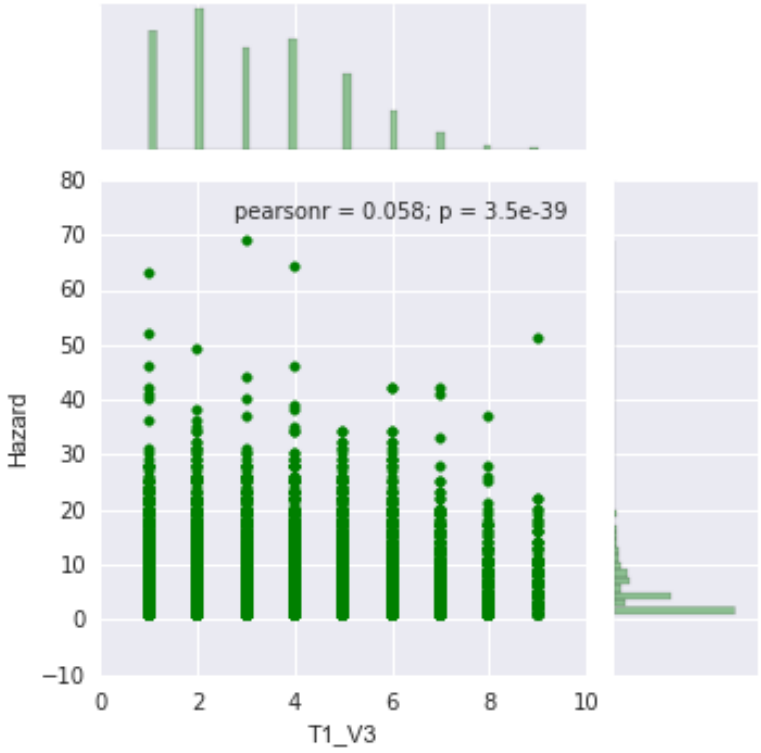
Based on the analysis in this report, the following conclusions and recommendations are made:

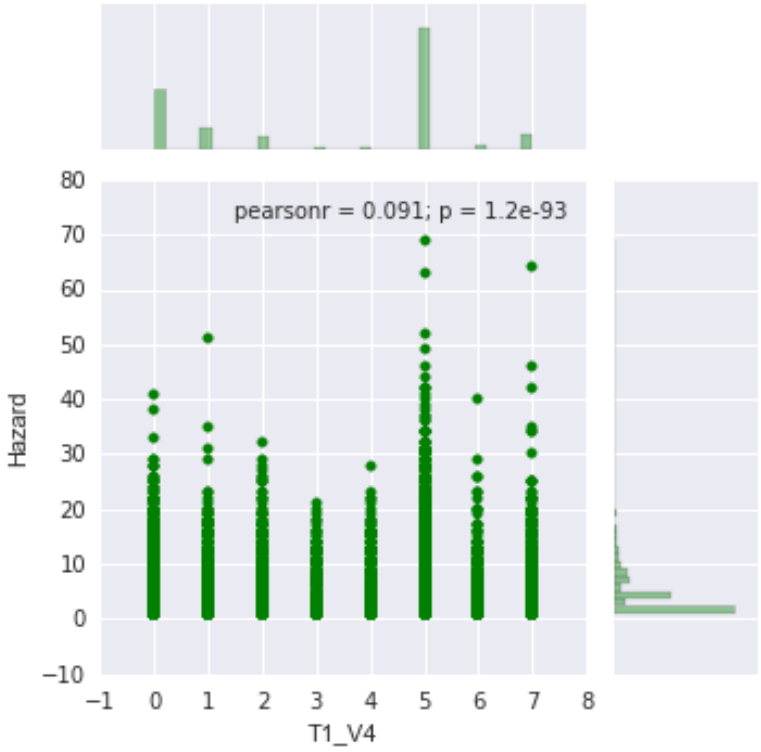
- ***The distribution of hazard scores appeared to follow an exponential probability distribution.***
- ***Tree-based ensemble methods such as Random Forests and XGBoost worked well for the property hazard data set.***
- ***Overall prediction accuracies were low. A business decision is required to determine whether the predictions are accurate enough to be useful.***
- ***Feature engineering, with domain-specific knowledge and known variables, may increase prediction accuracies. Feature engineering was difficult for this project given the anonymous features in the data set.***
- ***Ensembling models from diverse approaches proved to provide the best overall performance.***

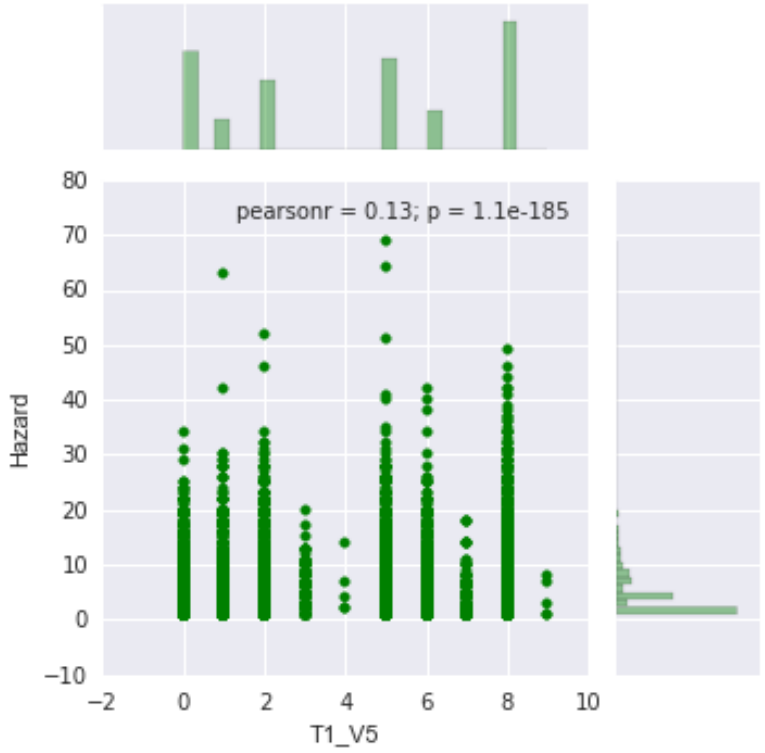
A. APPENDIX – JOINT PLOTS FOR ALL FEATURES

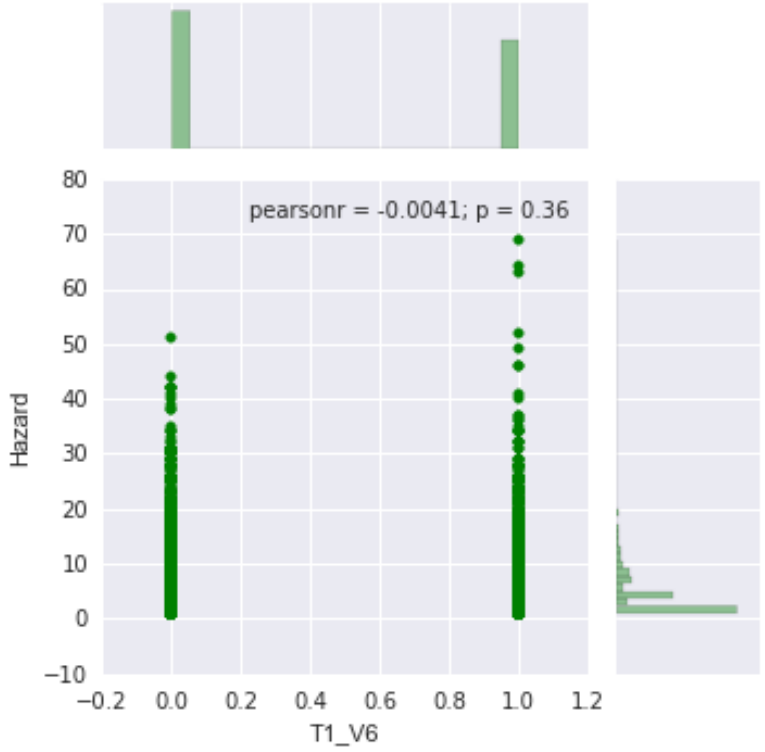


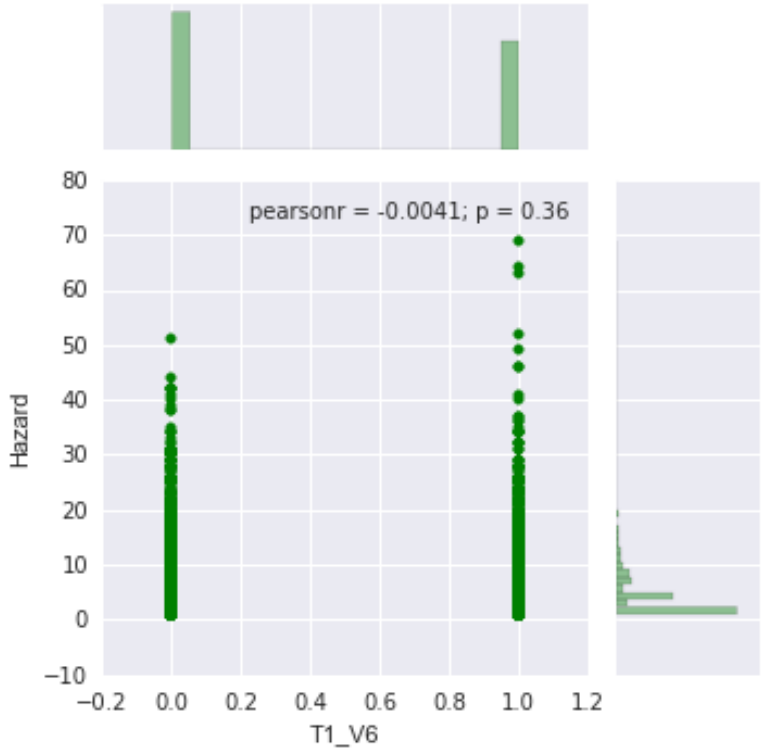


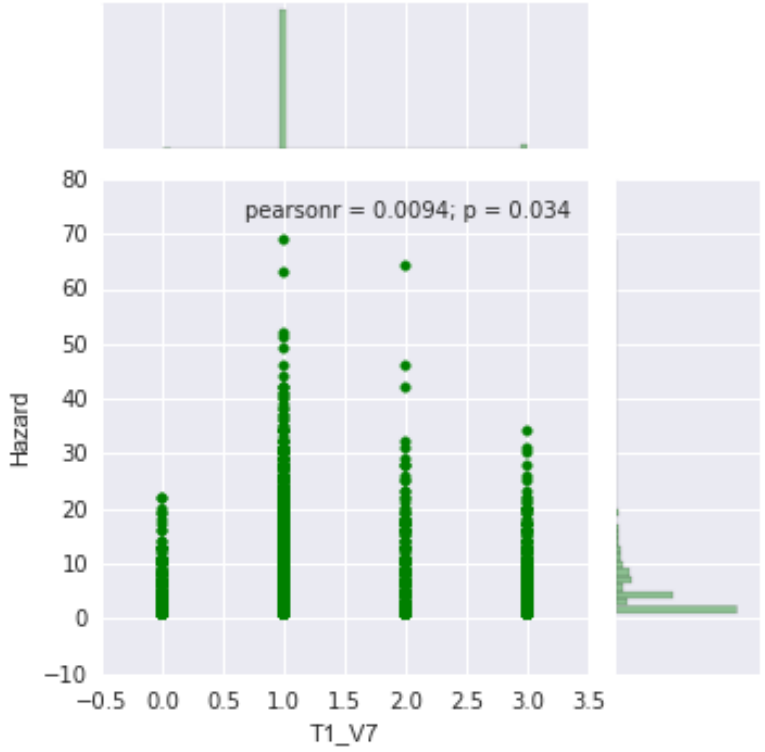

pearsonr = 0.058; p = 3.5e-39
Hazard
T1_V3
80
70
60
50
40
30
20
10
0
−10
0
2
4
6
8
10

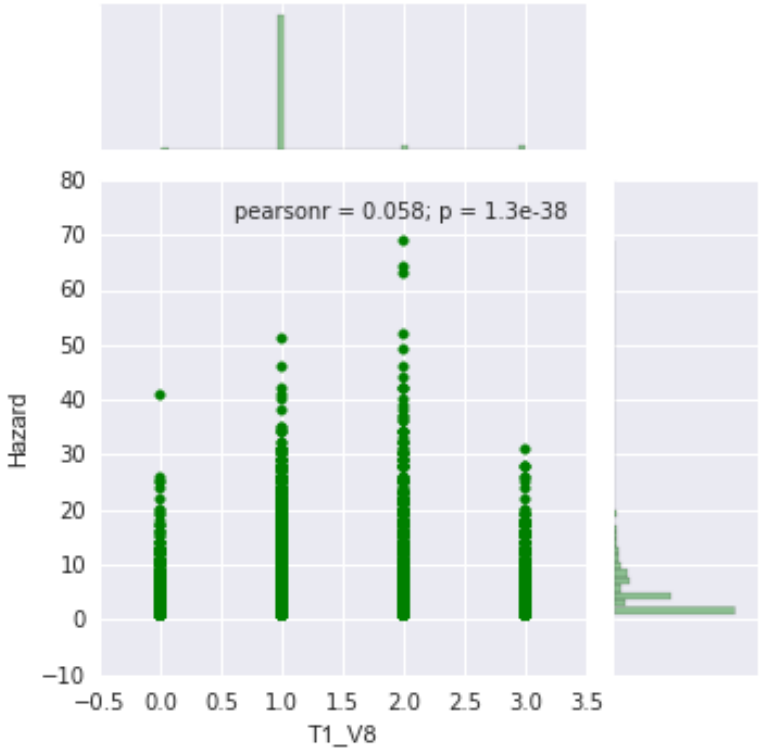

pearsonr = 0.091; p = 1.2e-93
Hazard
T1_V4
80
70
60
50
40
30
20
10
0
−10
−1
0
1
2
3
4
5
6
7
8

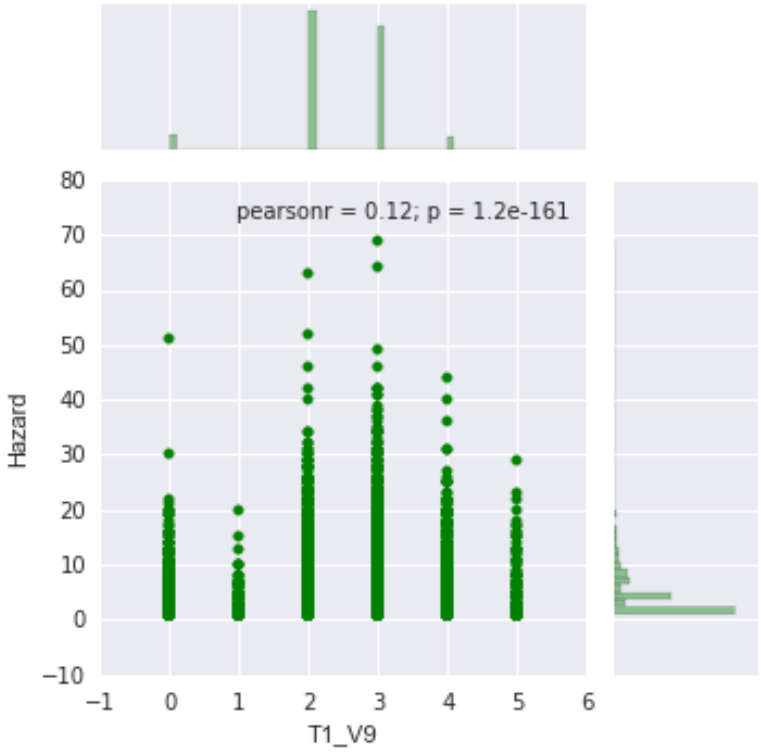

pearsonr = 0.13; p = 1.1e-185
Hazard
T1_V5
80
70
60
50
40
30
20
10
0
−10
−2
0
2
4
6
8
10

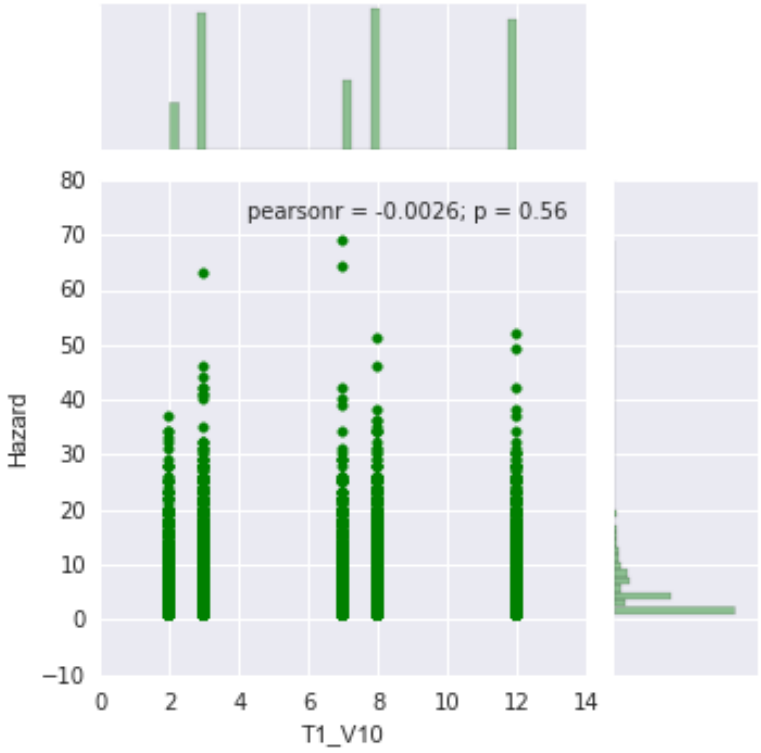

pearsonr = -0.0041; p = 0.36
Hazard
T1_V6
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

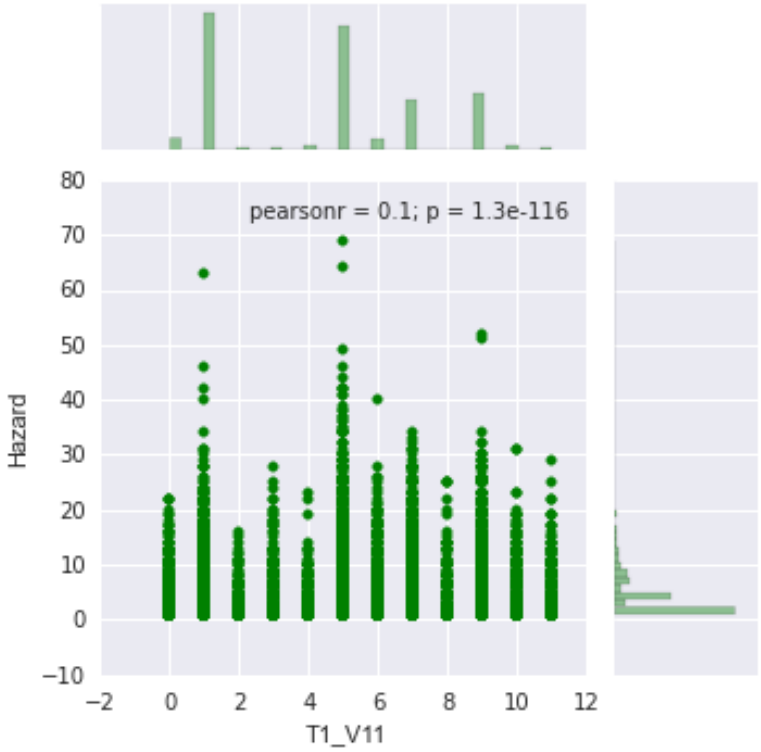

pearsonr = -0.0041; p = 0.36
Hazard
T1_V6
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

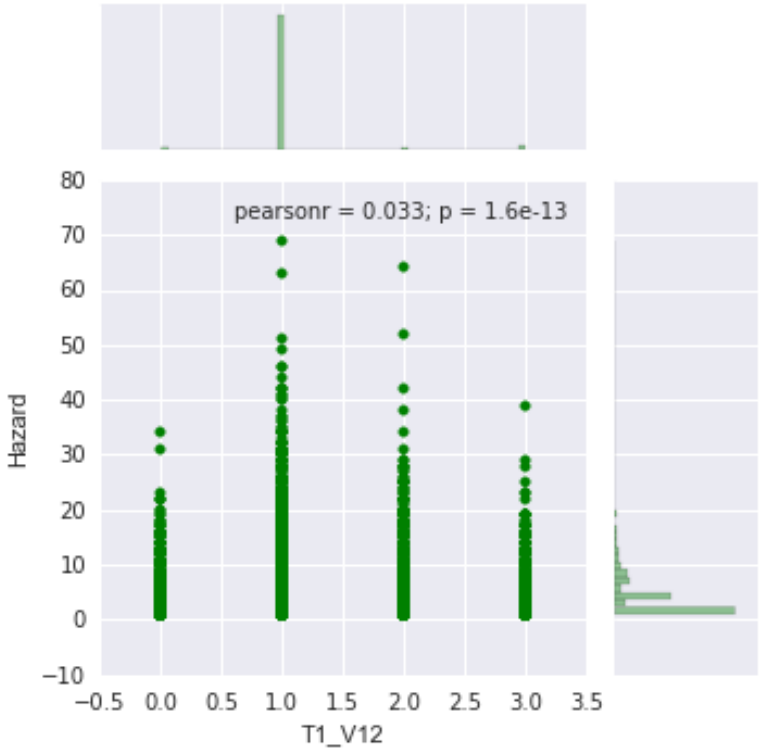

pearsonr = 0.0094; p = 0.034
Hazard
T1_V7
80
70
60
50
40
30
20
10
0
−10
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

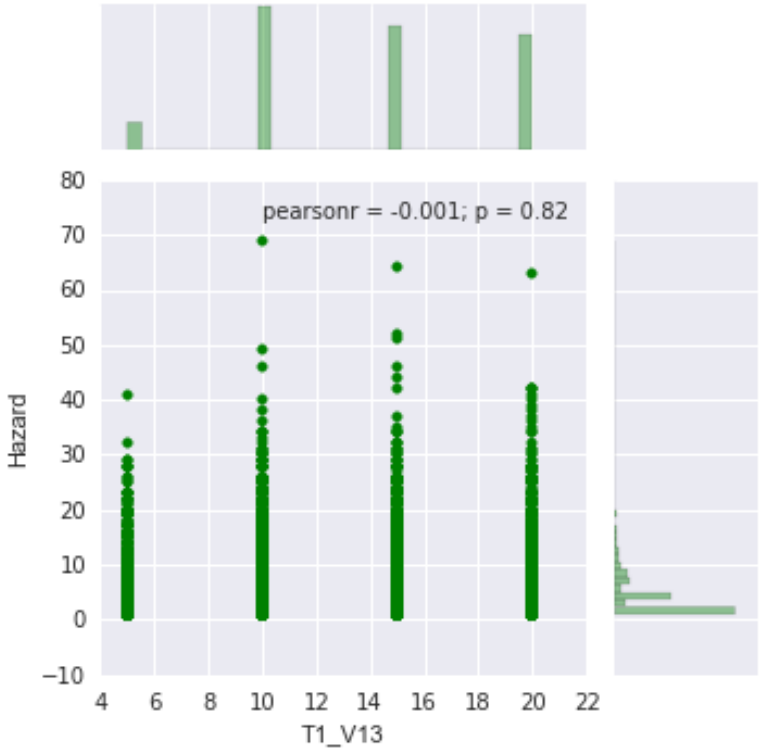

pearsonr = 0.058; p = 1.3e-38
Hazard
T1_V8
80
70
60
50
40
30
20
10
0
−10
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

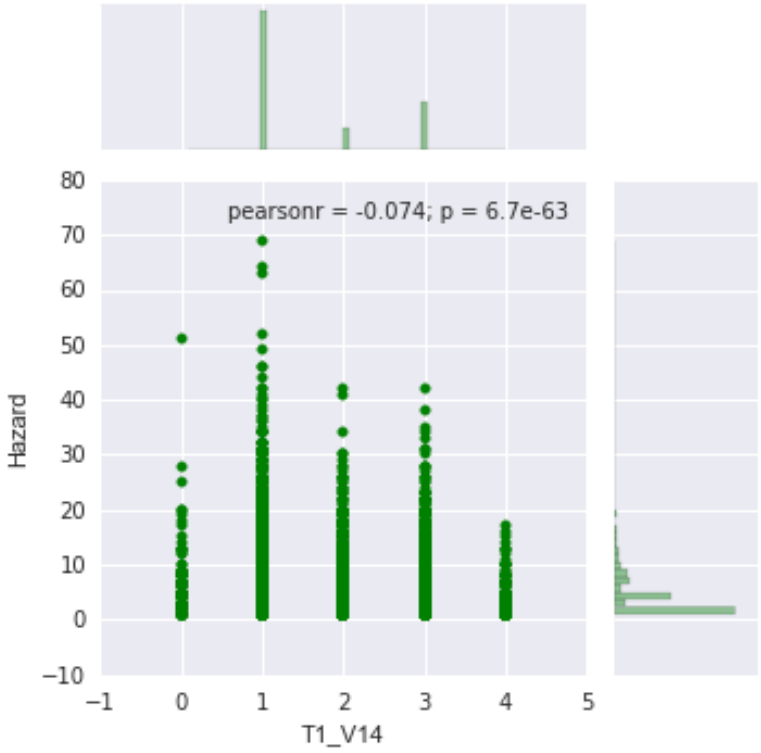

pearsonr = 0.12; p = 1.2e-161
Hazard
T1_V9
80
70
60
50
40
30
20
10
0
−10
−1
0
1
2
3
4
5
6

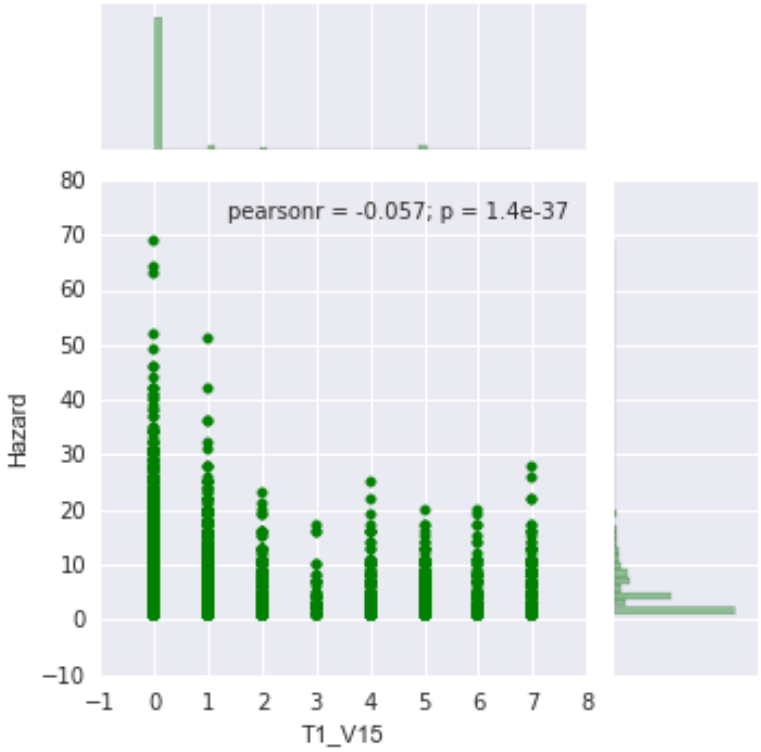

pearsonr = -0.0026; p = 0.56
Hazard
T1_V10

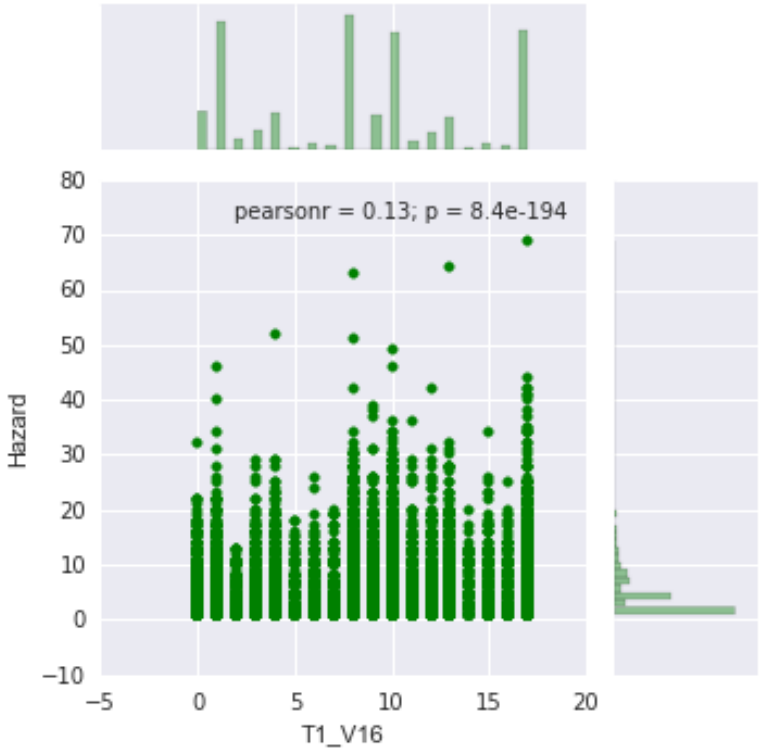

pearsonr = 0.1; p = 1.3e-116
Hazard
T1_V11

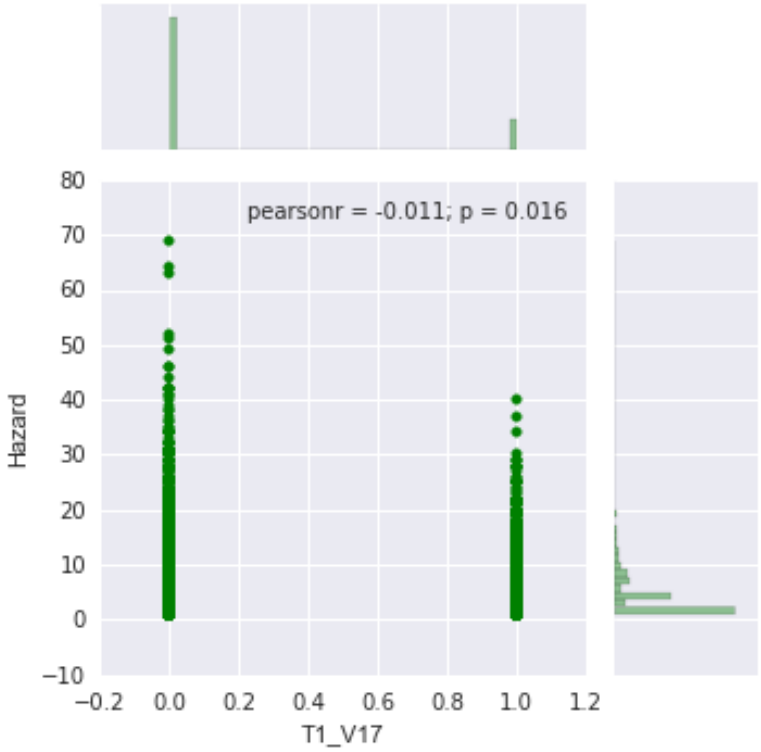

pearsonr = 0.033; p = 1.6e-13
Hazard
T1_V12
80
70
60
50
40
30
20
10
0
−10
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

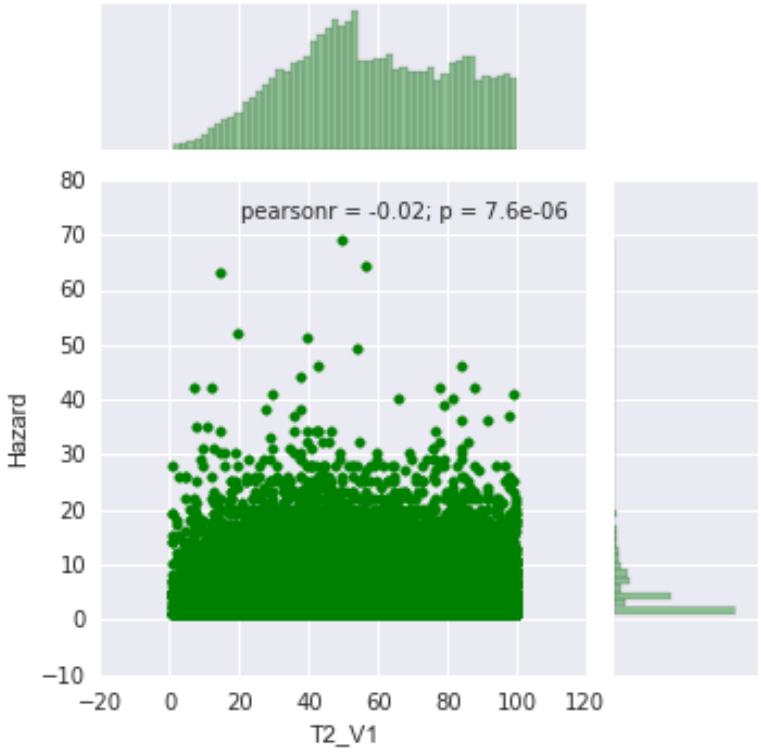

pearsonr = -0.001; p = 0.82
Hazard
T1_V13
80
70
60
50
40
30
20
10
0
−10
4
6
8
10
12
14
16
18
20
22

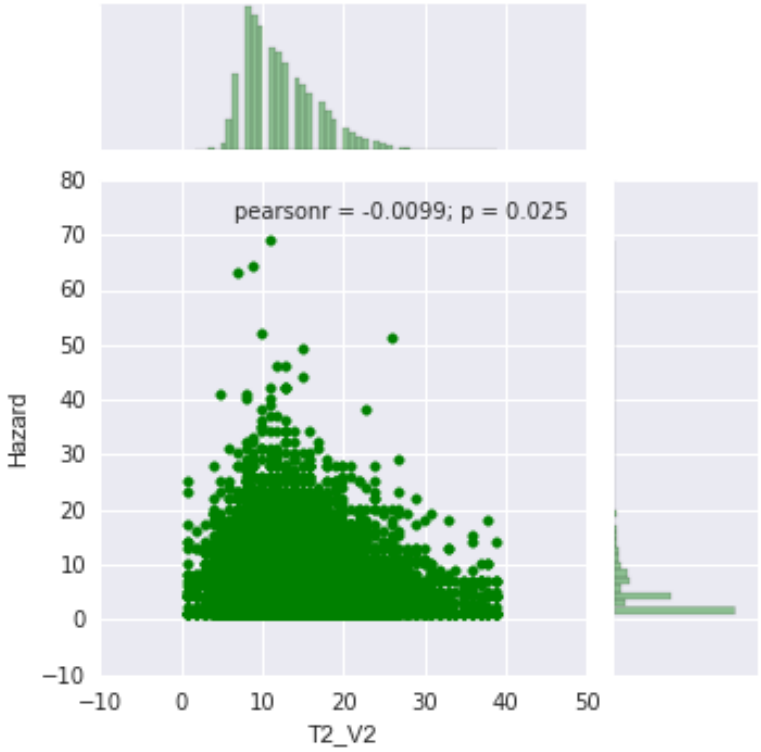

pearsonr = -0.074; p = 6.7e-63
Hazard
T1_V14
80
70
60
50
40
30
20
10
0
−10
−1
0
1
2
3
4
5

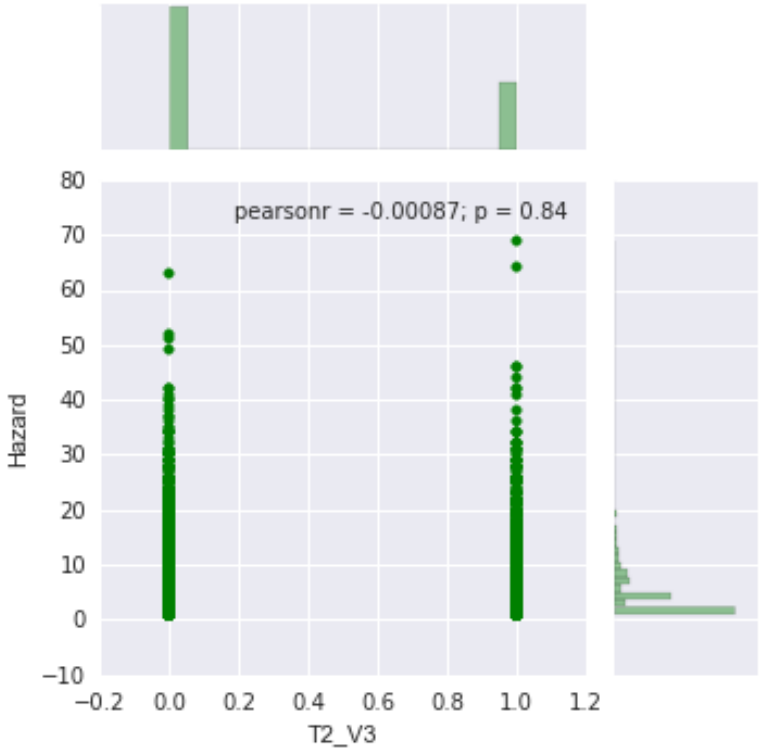

pearsonr = -0.057; p = 1.4e-37
Hazard
T1_V15
80
70
60
50
40
30
20
10
0
−10
−1
0
1
2
3
4
5
6
7
8

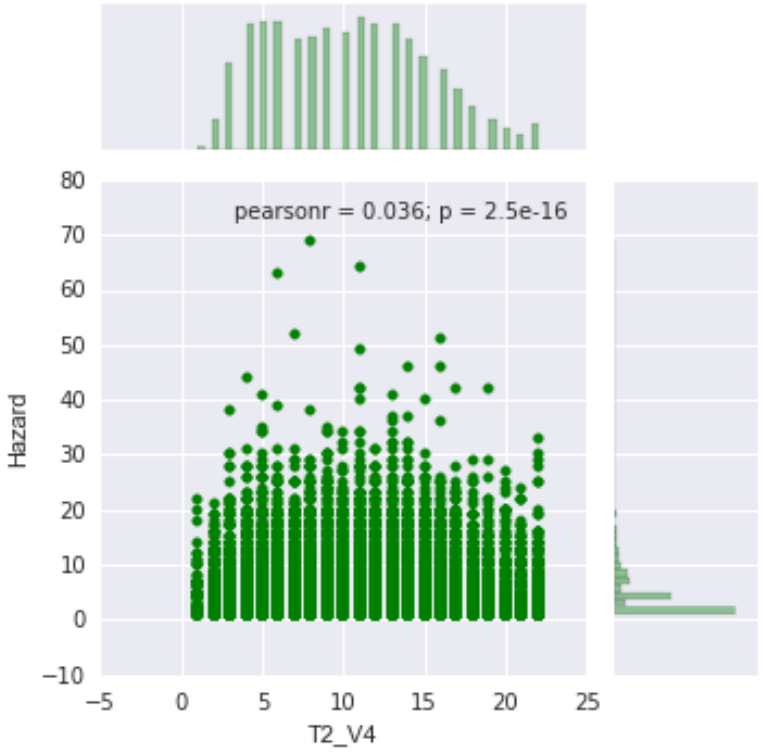

pearsonr = 0.13; p = 8.4e-194
Hazard
T1_V16
80
70
60
50
40
30
20
10
0
−10
−5
0
5
10
15
20

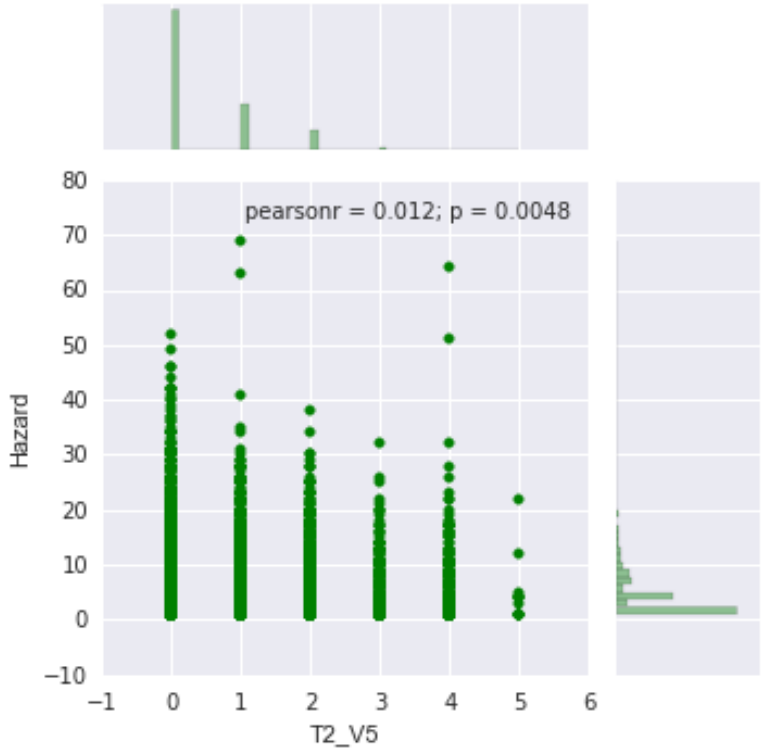

pearsonr = -0.011; p = 0.016
Hazard
T1_V17
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

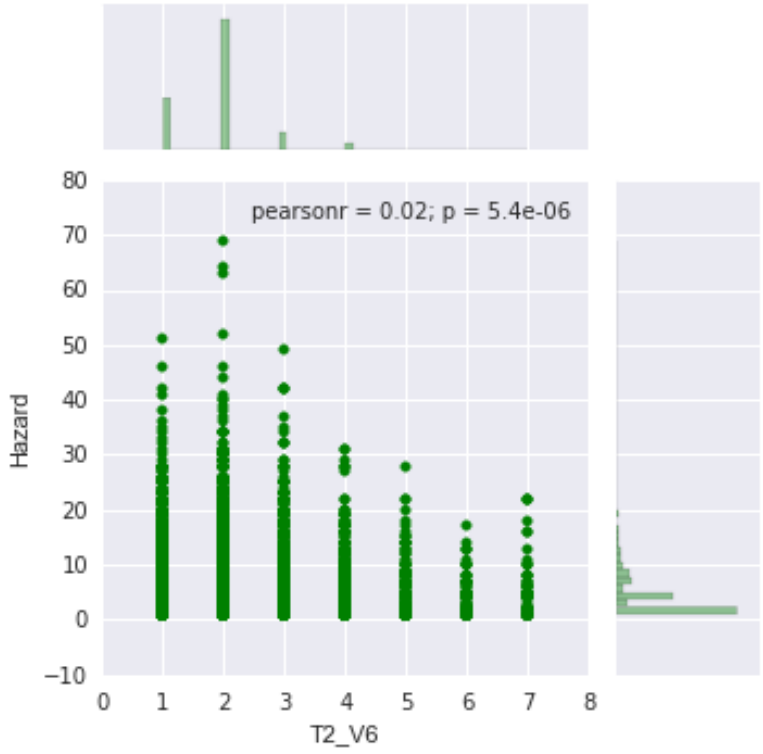

pearsonr = -0.02; p = 7.6e-06
Hazard
T2_V1
80
70
60
50
40
30
20
10
0
−10
−20
0
20
40
60
80
100
120

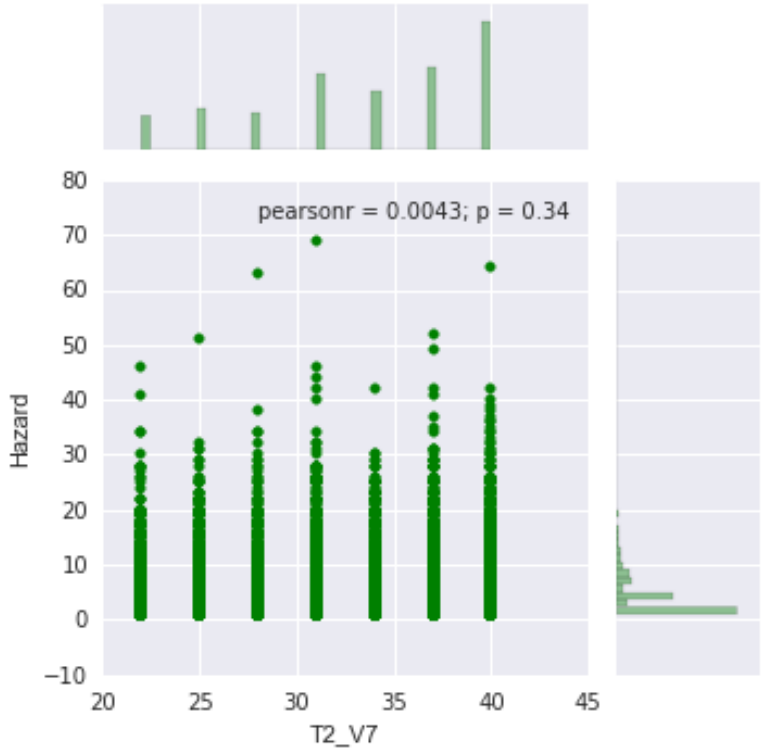

pearsonr = -0.0099; p = 0.025
Hazard
T2_V2
80
70
60
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50

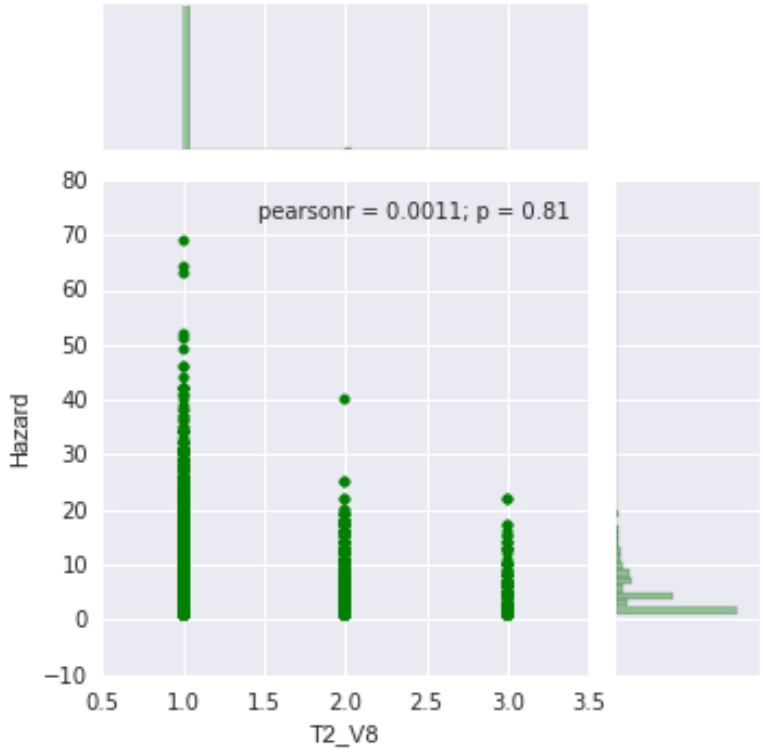

pearsonr = -0.00087; p = 0.84
Hazard
T2_V3
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

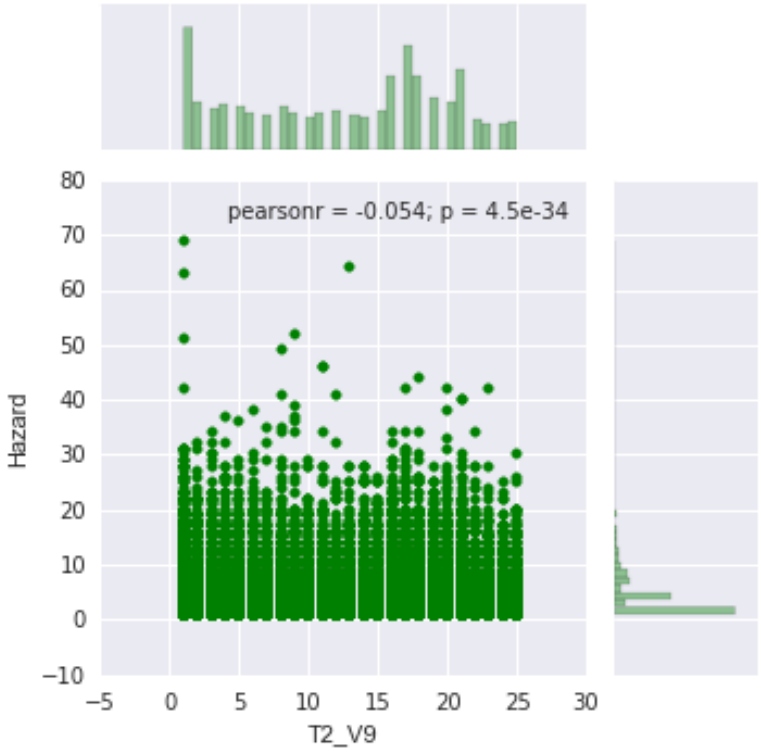

pearsonr = 0.036; p = 2.5e-16
Hazard
T2_V4
80
70
60
50
40
30
20
10
0
−10
−5
0
5
10
15
20
25

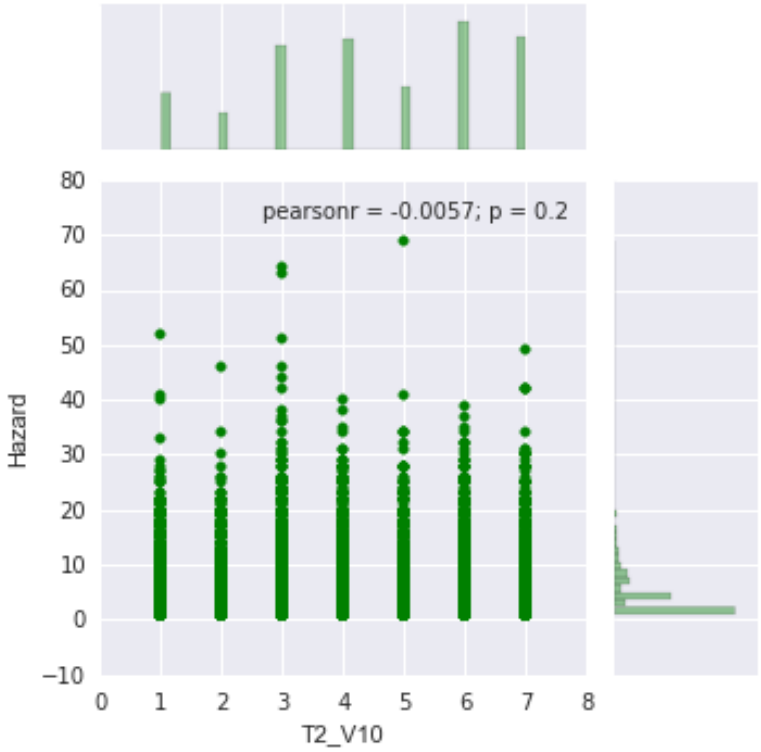

pearsonr = 0.012; p = 0.0048
Hazard
T2_V5

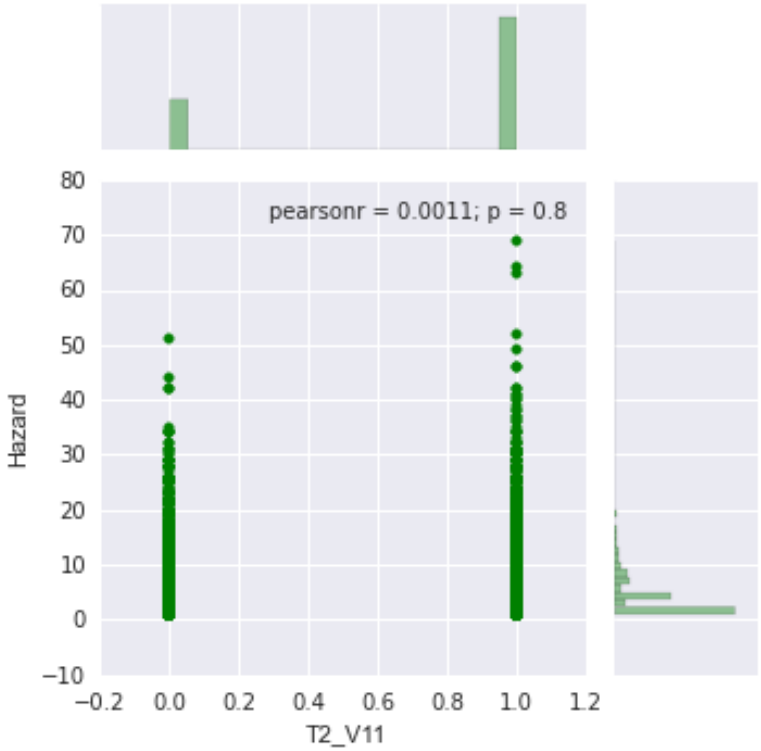

pearsonr = 0.02; p = 5.4e-06
Hazard
T2_V6

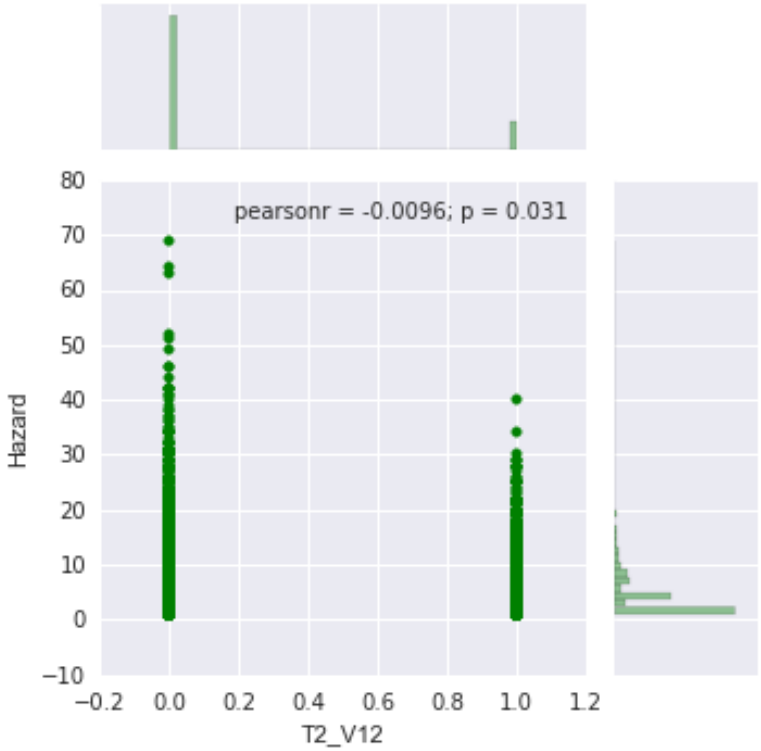

pearsonr = 0.0043; p = 0.34
Hazard
T2_V7
80
70
60
50
40
30
20
10
0
−10
20
25
30
35
40
45

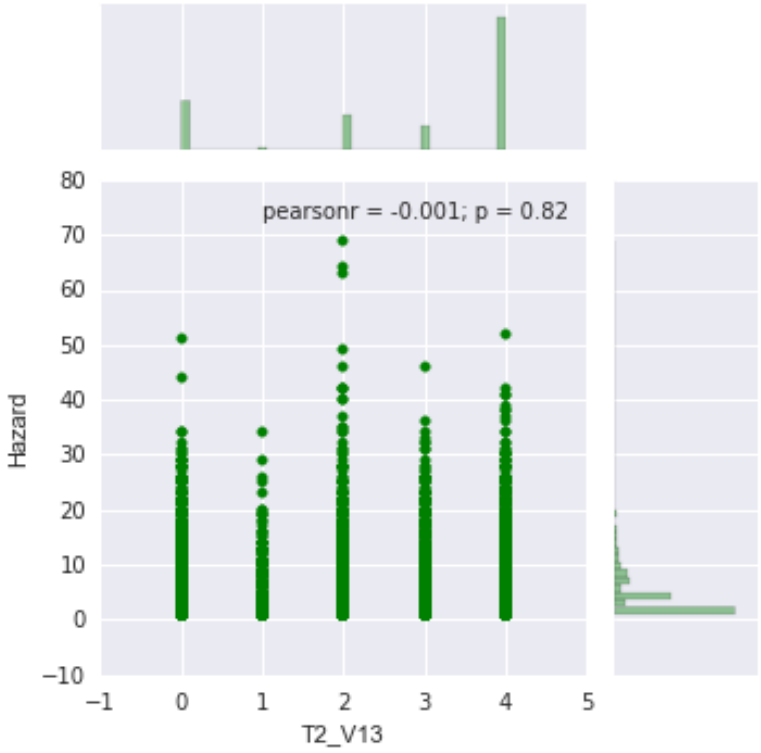

pearsonr = 0.0011; p = 0.81
Hazard
T2_V8
80
70
60
50
40
30
20
10
0
−10
0.5
1.0
1.5
2.0
2.5
3.0
3.5

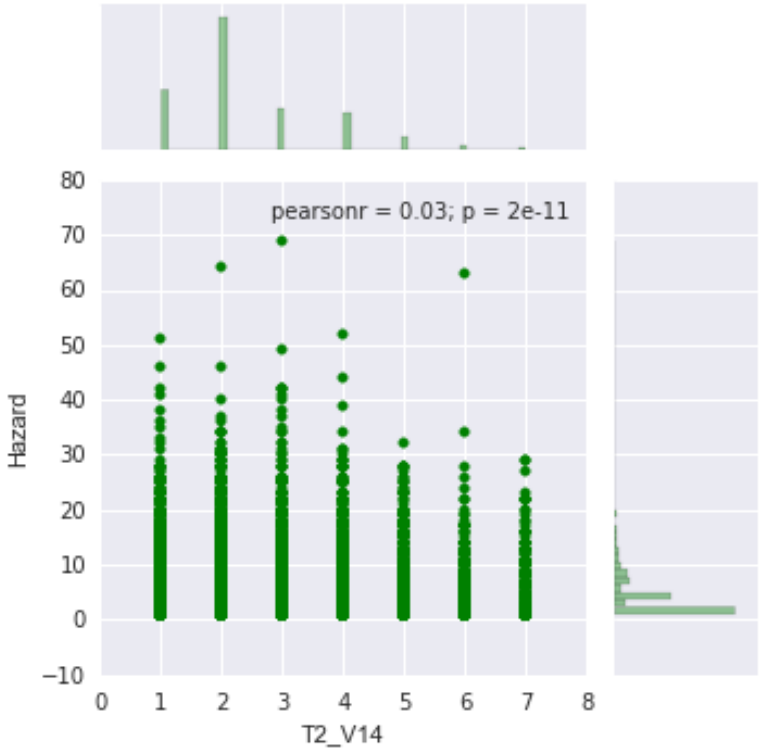

pearsonr = -0.054; p = 4.5e-34
Hazard
T2_V9
80
70
60
50
40
30
20
10
0
−10
−5
0
5
10
15
20
25
30

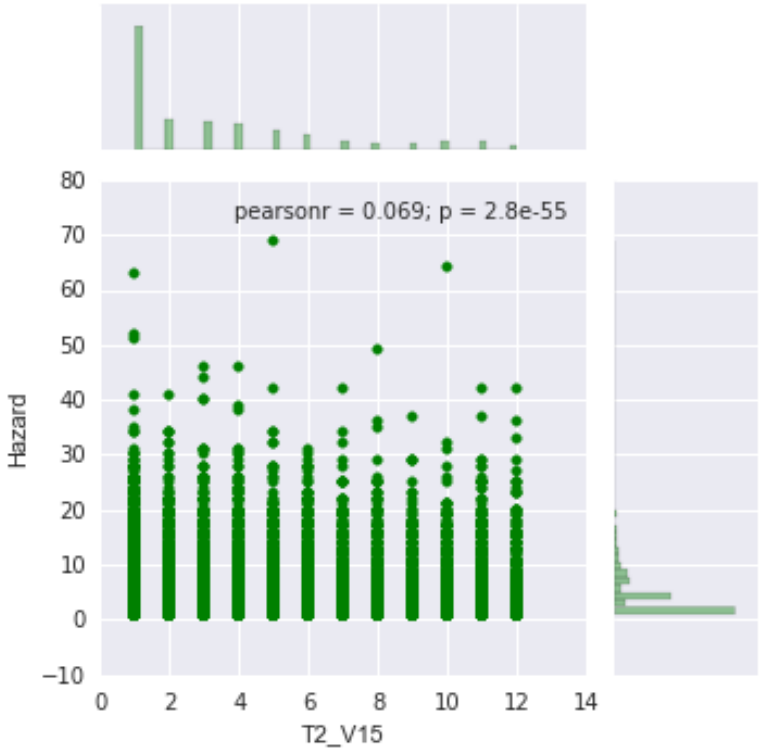

pearsonr = -0.0057; p = 0.2
Hazard
T2_V10
80
70
60
50
40
30
20
10
0
−10
0
1
2
3
4
5
6
7
8


pearsonr = 0.0011; p = 0.8
Hazard
T2_V11
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


pearsonr = -0.0096; p = 0.031
Hazard
T2_V12
80
70
60
50
40
30
20
10
0
−10
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


pearsonr = -0.001; p = 0.82
Hazard
T2_V13


pearsonr = 0.03; p = 2e-11
Hazard
T2_V14


pearsonr = 0.069; p = 2.8e-55
Hazard
T2_V15
80
70
60
50
40
30
20
10
0
−10
0
2
4
6
8
10
12
14