PRUDENT CHOICE

Build a model to help Prudential come up with a response to life insurance applicants

Poonam Rath
Data Science Final Project, 2016
General Assembly, San Francisco

DATA SOURCES

Kaggle (https://www.kaggle.com/c/prudential-life-insurance-assessment)

Prudential Life Insurance company

CONTENTS

1. Project goal
2. Analysis approach
3. Results
4. Conclusions
5. Next Steps

PROJECT GOAL

Analyze features collected from life insurance applicants provided by Prudential and build a model to predict the company's likely response.

Motivation for the problem:

Prudential wants to make it quicker and less labor intensive for customers to get a quote while maintaining privacy.

Status quo inefficient: takes the company a long time to come up with a decision, losing customers in the process.

If we come up with a model that accurately predicts the decisions the company comes up, the model can be incorporated to swiftly make decisions.

SIMILAR CASES

- Providing loans to individuals/businesses.
- Solution: models that analyze customer features and provide a credit score to help banks and financial institutions decide whether to provide a loan to an applicant.

ANALYSIS APPROACH

1. Data preparation
   - Overview of features and outcome
   - Feature engineering
   - Dealing with missing values
   - Normalization

2. Exploratory data analysis

3. Training, testing & cross-validation
   - Fit logistic regression, KNN, Random Forest and Gradient Boosting Classifier on a small sample of the dataset
   - K-fold cross-validation for multiple values of K
   - Learning curve to assess adequacy of data
   - Model performance evaluation (accuracy of prediction)
   - Feature selection
   - Test on out of sample test data set — get score from Kaggle

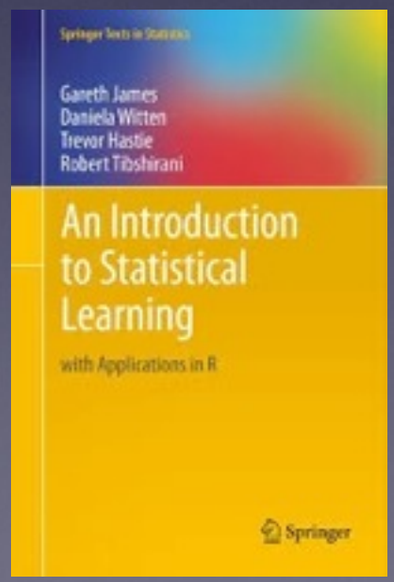
OVERVIEW OF DATA

- There are ~50,000 rows and 127 columns: 126 features and 1 outcome column. Outcome column is called “Response” and is nominal with 8 values (1,2,3,4,5,6,7,8).

- Most features are de-identified; we have a vague idea about what they might represent: “Product_Info”, “Employment_Info”, “Medical_keyword”.

- All but one feature columns are numeric.

RESOURCES USED FOR ANALYSIS



FEATURE ENGINEERING

- Obtained dummy variables for columns for which unique values were less than 0.5% of the total values.

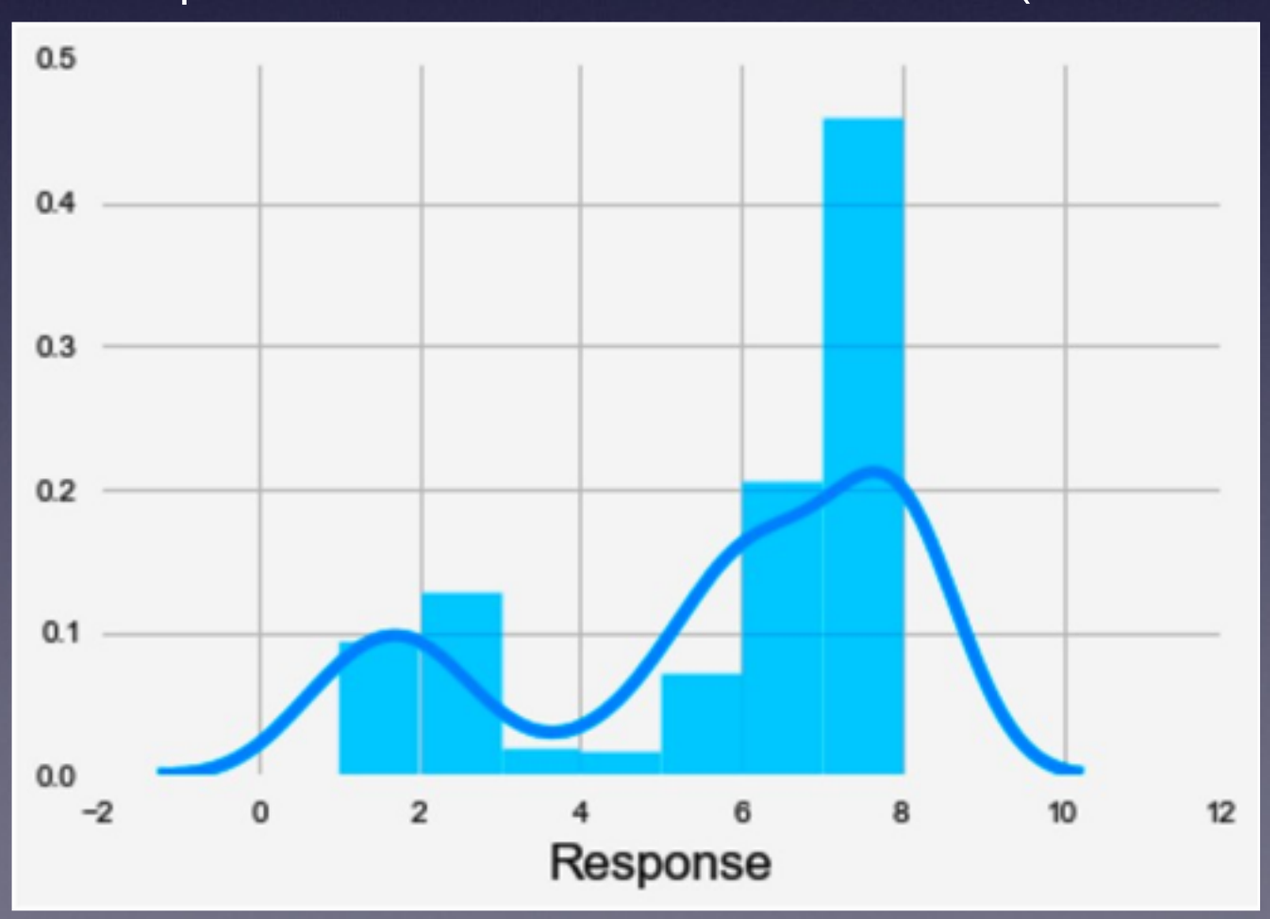
IMPUTING MISSING VALUES

- Imputed missing values with the median value of the columns.

FEATURE & RESPONSE DISTRIBUTIONS

- Distribution of the variables are normalized (according to Kaggle). If not, I would have looked at histograms, determined mean and stdev and scaled accordingly.

- Response column distribution (randomly sampled data: 500 rows)

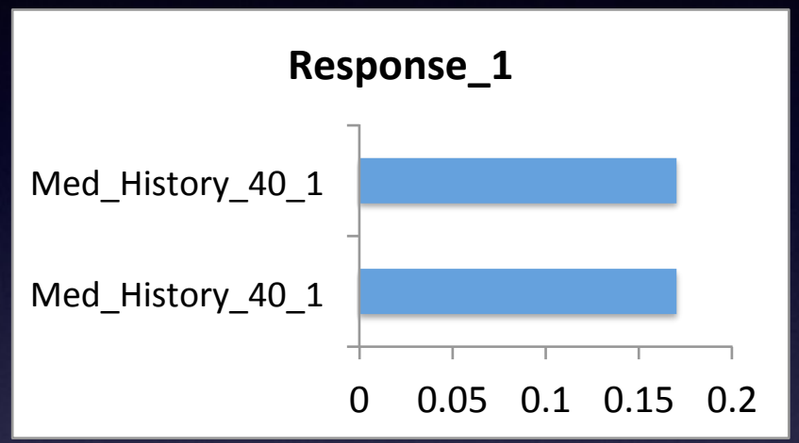


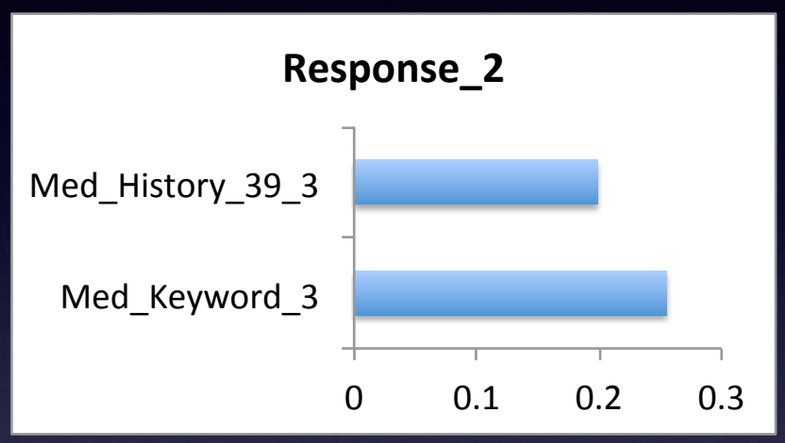
To consider:

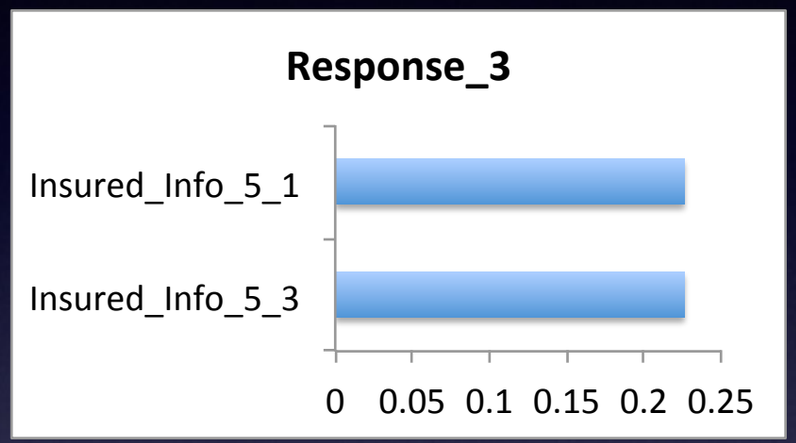
- SMOTE or upsampling #3,4
- Undersampling #7,8

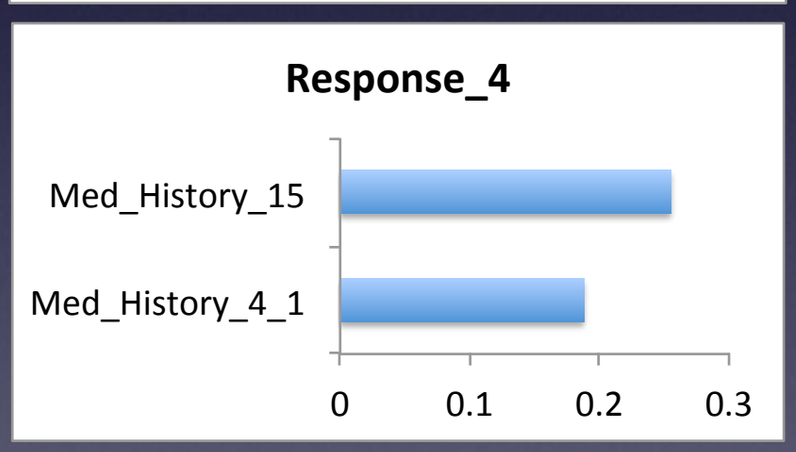
Exploratory data analysis -I

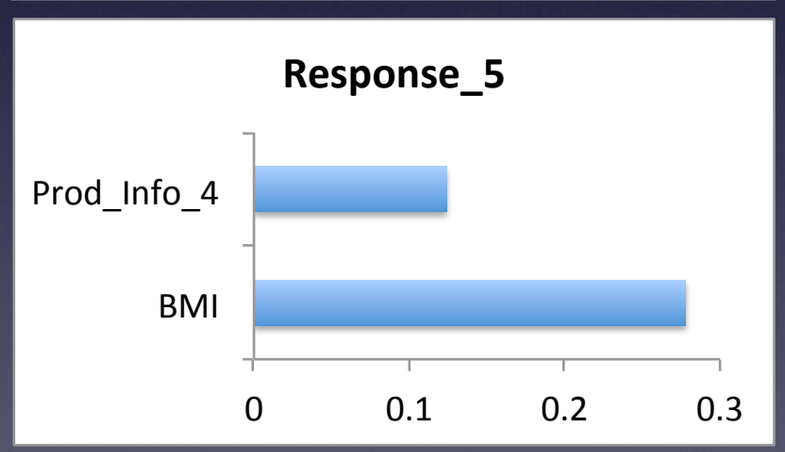
2) Most positively and negatively correlated feature for each response

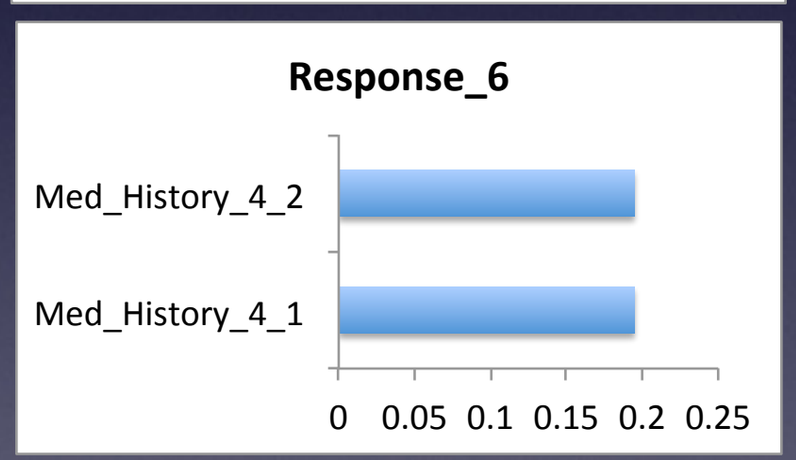


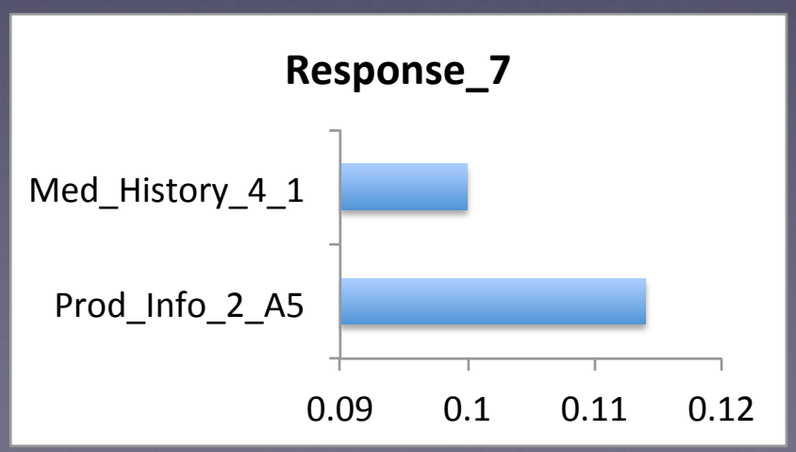


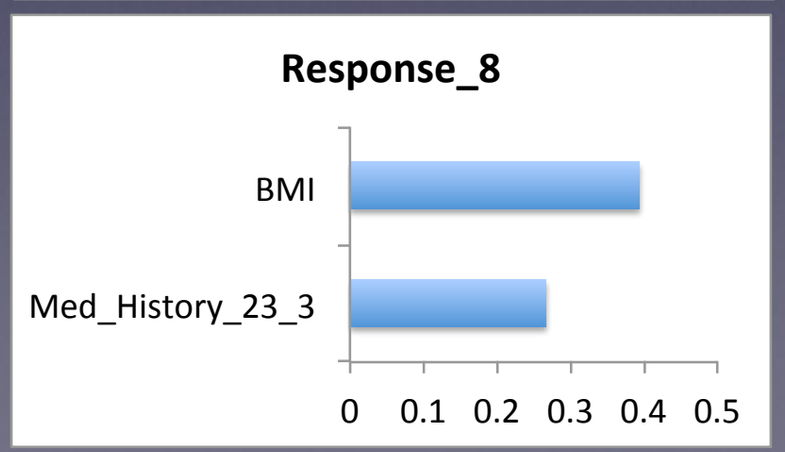






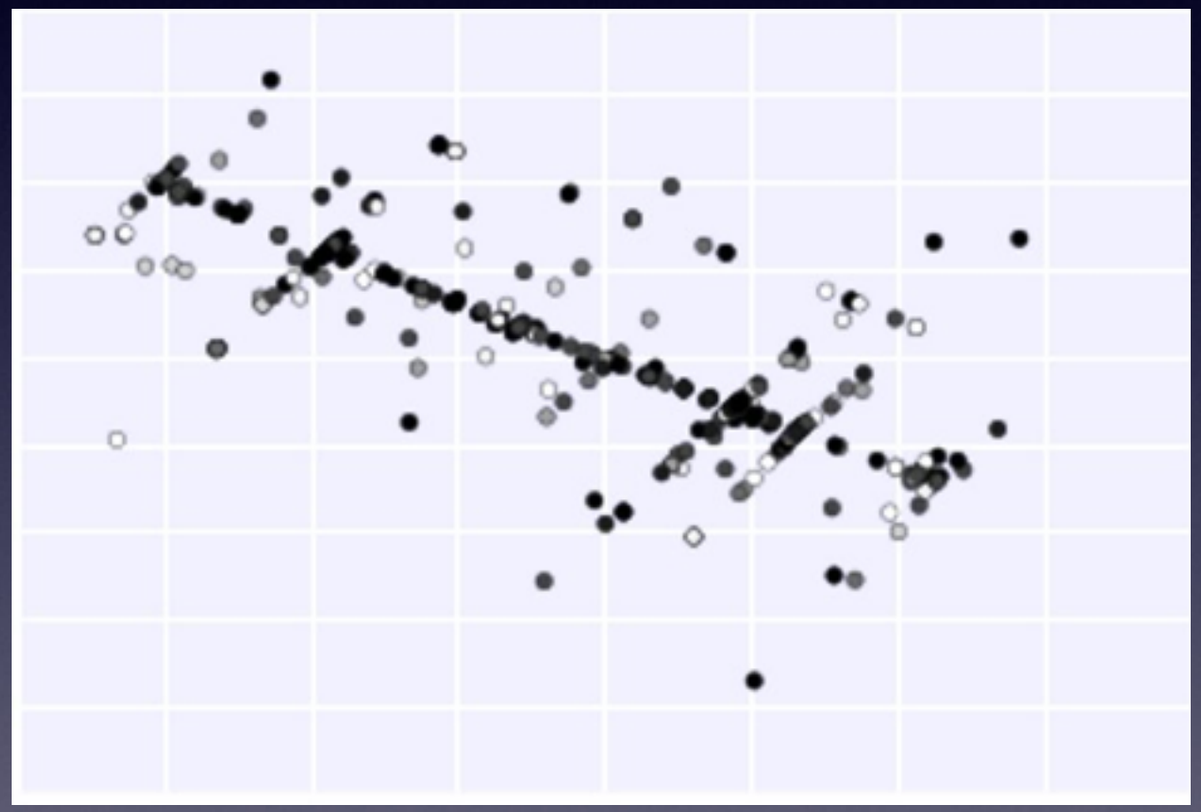






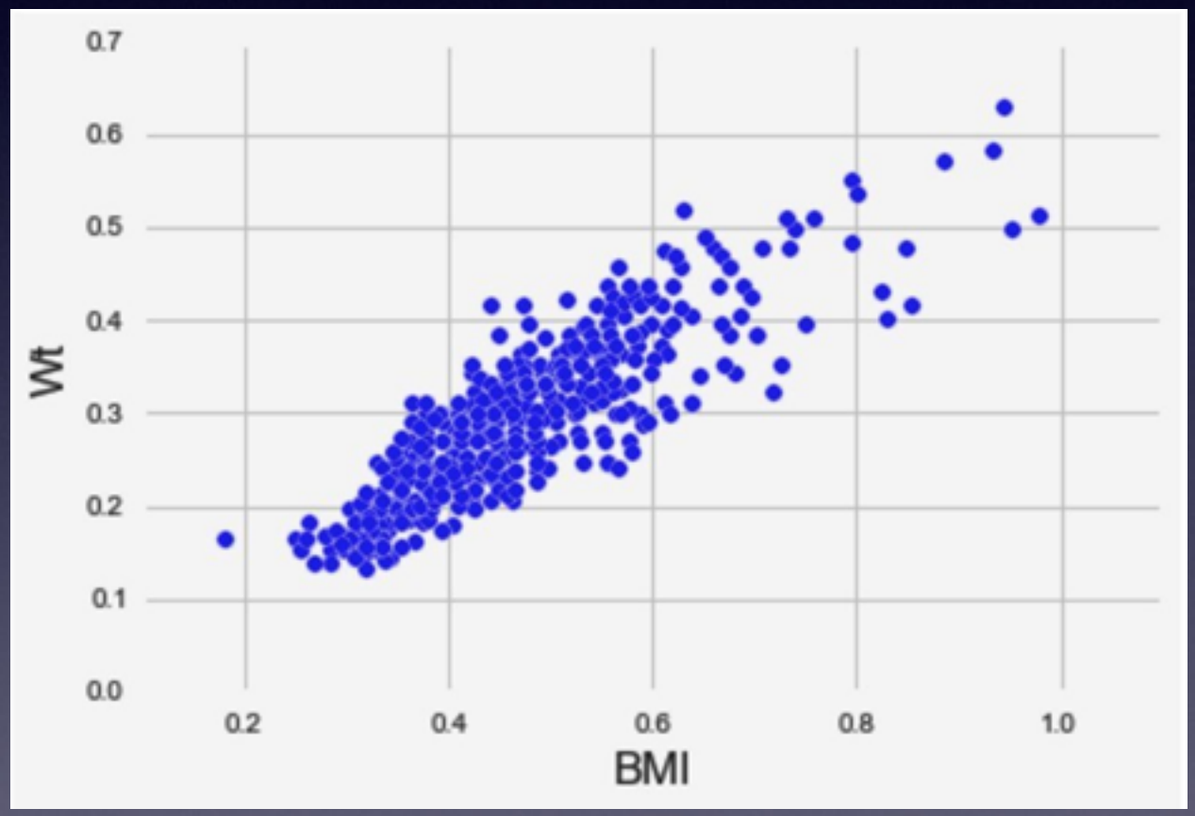
Exploratory data analysis -I

1) 2D projection of feature space using MDS, colored by *Response* shows that responses aren't clearly separable



Exploratory data analysis -II

BMI is positively correlated with Weight (subset of data)



Training a model - First Pass

- Using all features on a small sample of the data, I fitted the following models:
    - Random Forest Classifier
    - K-Nearest Neighbor Classifier
    - Logistic Regression
    - Gradient Boosting Classifier

- K-fold cross-validation score (Insert graph).
- Learning curve shows that we have enough data.

Measuring Model Performance

- My choice of model performance metric was “accuracy” (% of times the model guessed correctly).
    - Logistic Regression: 33% accuracy
    - K-Nearest Neighbor Classifier: 21%
    - Naive Bayes: 19%
    - Gradient Boosting Classifier: 46.6% accuracy
- For this competition, Kaggle evaluates submissions on a different metric: weighted kappa. Though there are ways to incorporate custom scoring metrics, I left this out for a later stage since I wanted to focus on getting my model predictions to be as accurate as possible.

Next Steps

- Make Response column more even : SMOTE/undersample
- Run the models on the entire dataset on Amazon’s EC2 cloud server.