Model Prediction Interpretation with Shapley Additive Explanations

ODSC East – May, 2019

Applied Statistics Seminar
Andrew Dodge
Shusaku Asai
10/17/2019

Agenda

- Introduction / Motivation
- Shapley Values
- Strengths of SHAP
- Feature Importance in xgboost
- Python example – King County housing
- R example – Control Tower

Introduction

- Modern machine learning methods often achieve highest prediction accuracy at the cost of low interpretability.
- Methods such as LIME and DeepLIFT are known to provide explanations for complex models.
- SHAP offers a novel, unified framework to understand why a model makes a certain prediction and its feature importance.

Properties of Strong Feature Importance Measures

So, what makes a good feature attribution method?

- **Consistency** – Feature attribution remains consistent with changes in the model.
  - If a model is changed such that it relies more on a feature, its importance should not decrease.
- **Accuracy** – The sum of all feature importances should sum to the overall importance of the model.
  - Allows for comparison of variable importances relative to each other.

Shapley Values & Game Theory

- Given a coalitional game of multiple players, shapley values estimate the fairest way to distribute payoff based on each player's marginal contribution.

What does this have to do with Machine Learning?

- The same principle can be applied to estimate the marginal contributions of each feature/variable to model predictions.

Shapley Value Equation

$$\varphi_j(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!\,(p-|S|-1)!}{p!} \left(val\left(S \cup \{x_j\}\right) - val(S)\right)$$

- φ_j = Shapley value for feature *j*
- S = all possible combinations of features excluding feature *j*
- x_j = feature *j*

Both consistent and accurate

Shapley Values Explained

- Castle building – 3 contributors
  - Each work in order
  - What is each person's contribution?
  - What is each person’s value to the team?
- Can be dependent on the order

- Average of the marginal contributions across all permutations

Shapley Values and SHAP

- Shapley values become computationally expensive as your dataset grows.
- SHAP provide estimates of shapley values, allowing for quick computation in practice.
- This is achieved by leveraging the hierarchy of tree based models.

SHapley **A**dditive ex**P**lanations

- For a single prediction, SHAP provides each model feature an importance value

- SHAP strengths:
    1. Measure global importance, uniquely
    2. Measure local importance
    3. Can be used on any tree based model

Gradient Boosting Machine

- Tree based ML algorithm, where an ensemble of trees/regression models are fit successively.
- ‘xgboost’ is a popular GBM tool that can be loaded into R (famous for winning ML competitions)
- ‘xgb.importance’ provides multiple measures of feature importance, but are difficult to interpret.

Python Example - Background

- King County (WA) housing prices
  - Homes sold between May 2014 – May 2015
  - 21,613 observations by 21 columns
  - “price” as outcome
  - Variables such as “bedrooms”, “bathrooms”, “waterfront”, “yr_built”, “sqft_living”.
- Split data 80 – 20
- Built model with RandomForestRegressor in python

Python Example - Background

- King County, WA
  - Includes the Seattle area, east to the Cascades

Setup Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
import shap
```

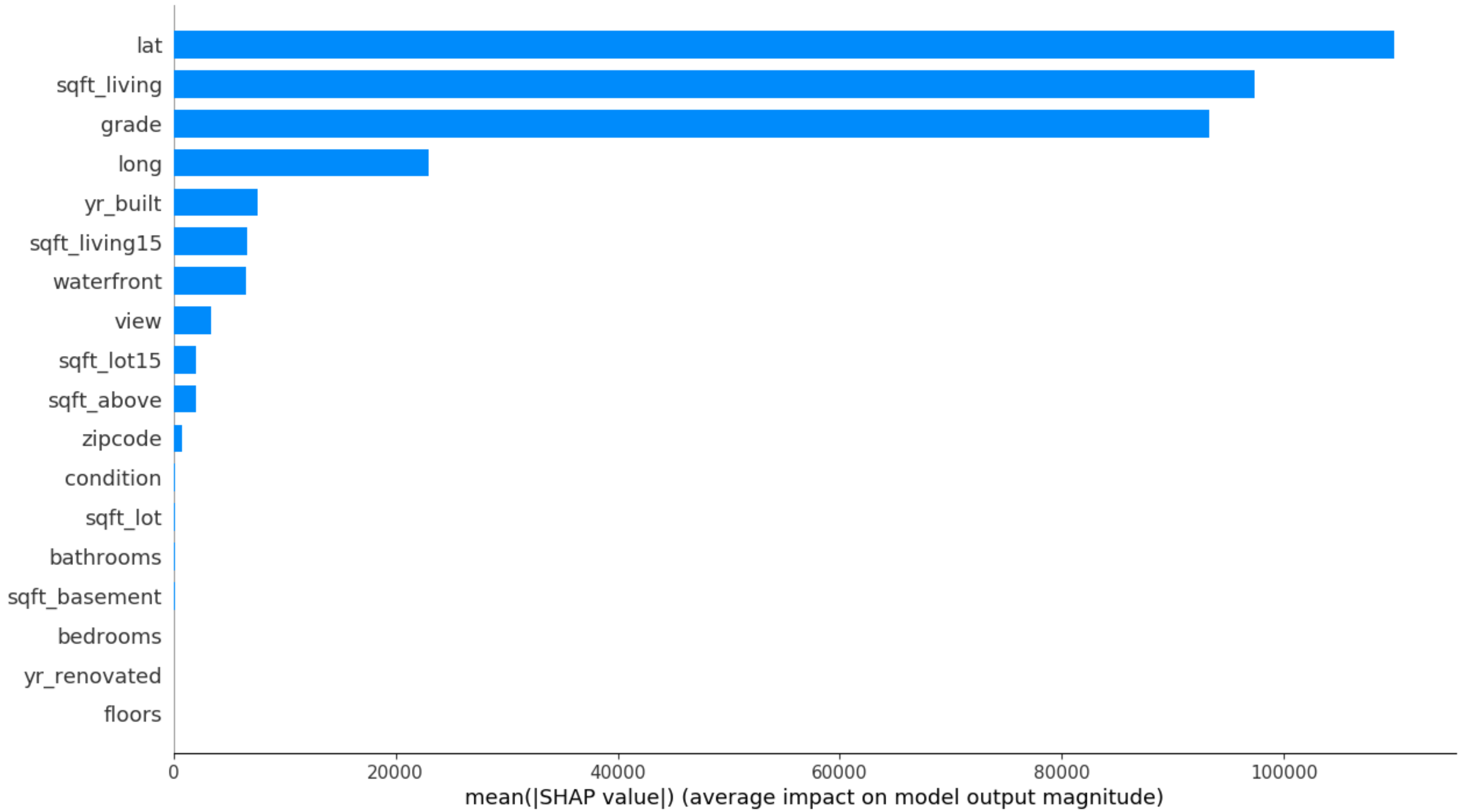
Setup Code

```python
df = pd.read_csv('kc_house_data.csv')

# The target variable is 'price'.
Y = df['price']
X =  df[['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15']]

# Split the data into train and test data:
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)

# Build the model with random forest regression:
model = RandomForestRegressor(max_depth=6, random_state=0, n_estimators=10)
model.fit(X_train, Y_train)

shap_values = shap.TreeExplainer(model).shap_values(X_train)
```

SHAP Summary Plot

- Global variable importance

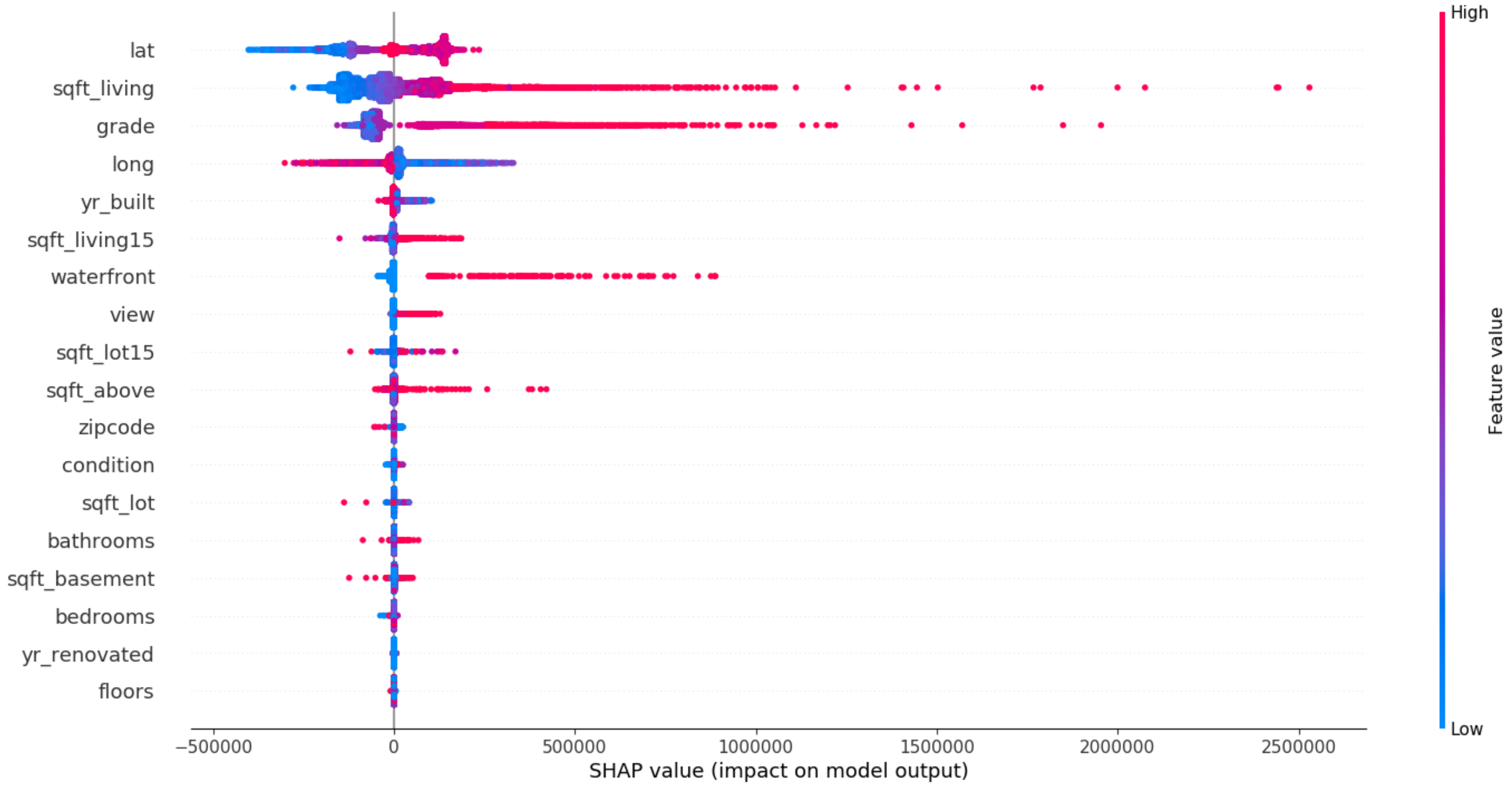
```
33    shap.summary_plot(shap_values, X_train, plot_type="bar")
```



SHAP Summary Plot

- Displays feature importance, impact, original value, correlation

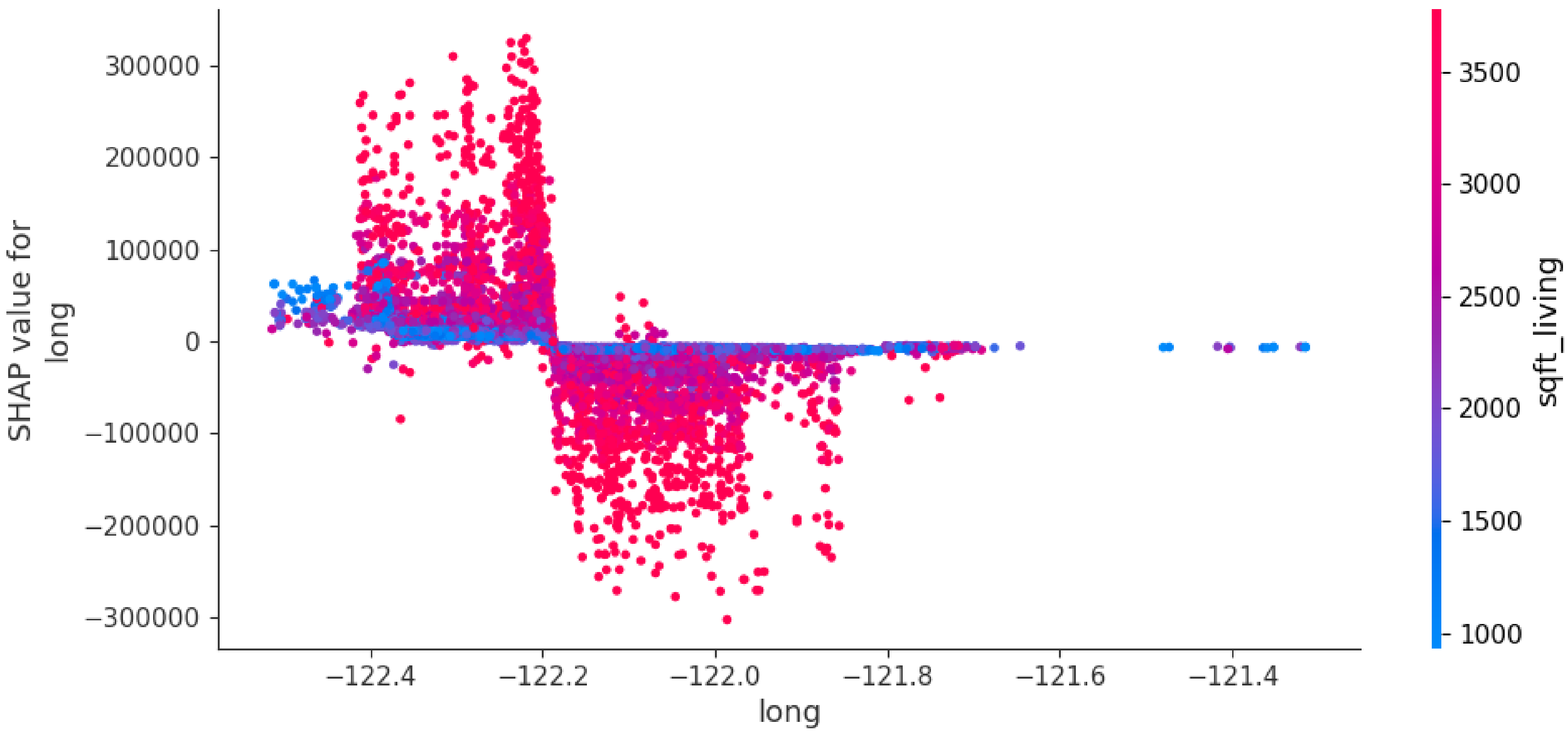
```
35    shap.summary_plot(shap_values, X_train)
```



Partial Dependence Plot

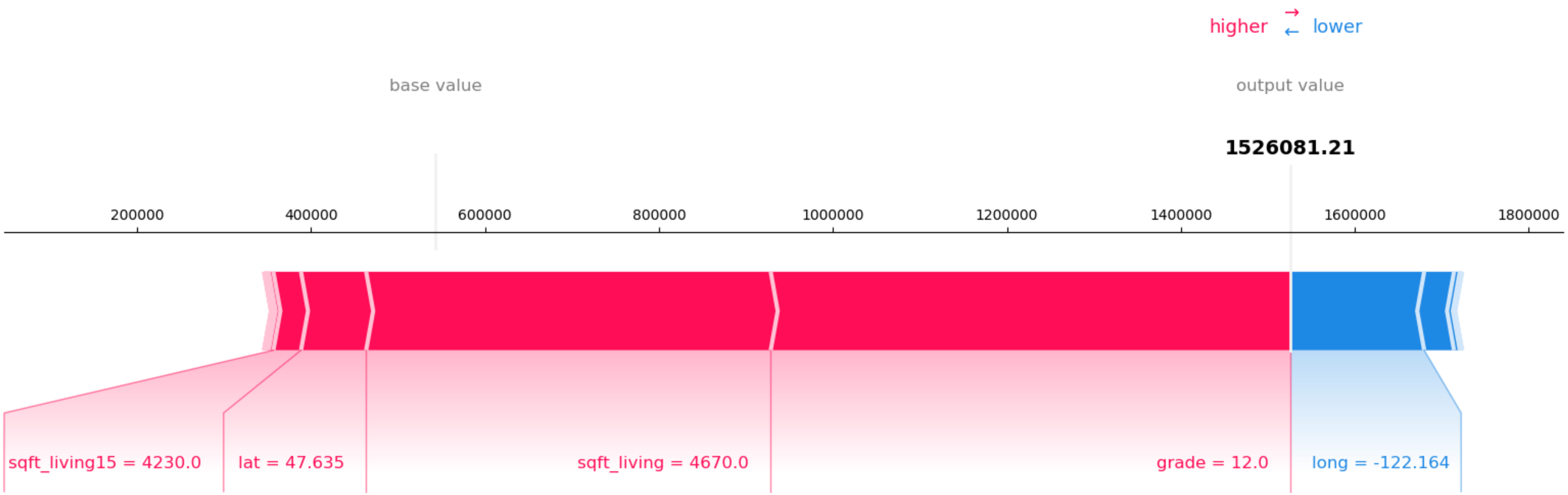
- Relationship between 'price' and 'long', as well as the variable it interacts most with

```
36   shap.dependence_plot("long", shap_values, X_train)
```



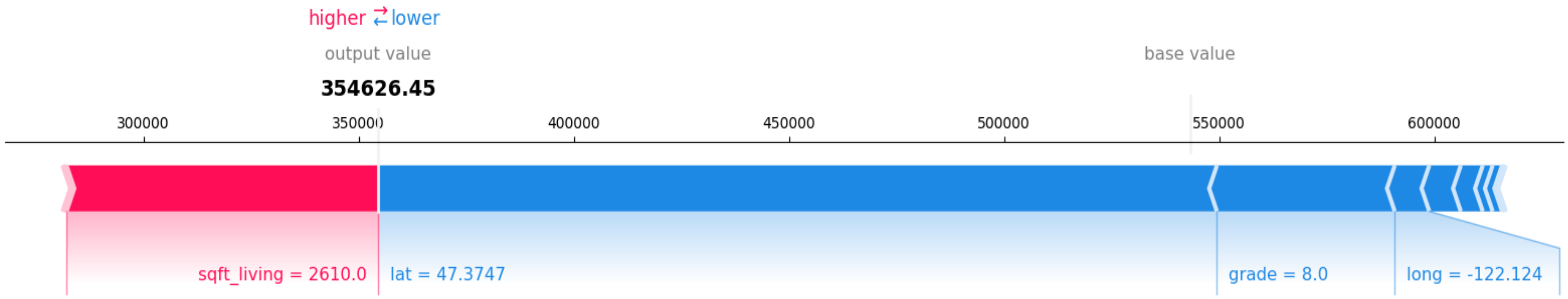
Prediction Plot

- Local interpretability
  - Output value, base value, top predictor impact
  - Actual value = $1,578,000



Prediction Plot

- Local interpretability
  - Actual value = $345,000



Example in R - Control Tower

- Outcome: Palliative Care Consultation
- ~150 variables

  Demographics, Prior History (inpatient visit history, prior palliative care consultations), existing conditions (~70 HCC categories), laboratory values, nursing unit location
- ~705,000 rows
- ~70,000 encounters
- ~2,400 PC consults (outcome = “pc”)

Set up

- This example uses R 3.6.1

````
```{r, echo=FALSE, message=F, warning=F}
library(xgboost)
library(tidyverse)
library(Matrix)
library(data.table)
library(rsample)
```
````

Data Split: Testing and Training

```
#read in your data:
mydata <- readRDS(file="~/private/SHAP_data.rds")

#split the data into train and testing
set.seed(312)

#use the 'rsample' package to prepare for testing and training split
first_split <- initial_split(mydata,prop = 8/10, strata = "pc")

#create training and testing data
ptrain <- training(first_split) #80% training
ptest <- testing(first_split) #20% testing/validation
```

Create a Sparse Matrix

```
#Copy the training data to a new object
ptrain_lab <- ptrain
#Remove the outcome variable
ptrain_lab$pc <- NULL

#change all chr to factor with an 'unclass' statement.
data_train <- as.data.frame(unclass(ptrain_lab))

#turn the data into a matrix
data_train_mat <- as.matrix(data_train, nrow = 564156, ncol = 130)

#create a sparse matrix
sparse_matrix <- Matrix(data_train_mat, sparse = TRUE)
train_matrix <- xgb.DMatrix(data = sparse_matrix, label = ptrain$pc)
```

Modeling the Data

```r
#Gradient Boosting Machine
gbm_model <- xgboost(data = train_matrix, max.depth = 2,
               eta = 1, nthread = 2, nrounds = 2,
               objective = "count:poisson", verbose = 2)
```

Test Set: Sparse Matrix

```
#copy the test data to a new object
ptest_lab <- ptest

#remove the outcome variable from the copied data
ptest_lab$pc <- NULL

#change character variables to factors with 'unclass'
data_test <- as.data.frame(unclass(ptest_lab))

#create a sparse matrix
data_test_mat <- as.matrix(data_test, nrow = 141038, ncol = 130)
sparse_matrix_test <- Matrix(data_test_mat, sparse = TRUE)
```

Generate SHAP Scores

```
#GBM predict on the test dataset
pred <- predict(gbm_model, sparse_matrix_test, shap)

#Create the SHAP scores
shap_scores <- shap.score.rank(xgb_model = gbm_model,
                               shap_approx = TRUE,
                               X_train=sparse_matrix)

#Output the Global top 10 important variables into a dataframe
shap_scores_std <- shap.prep(shap  = shap_scores,
                             X_train = sparse_matrix,
                             top_n = 10)

#Plot the top 10 variables in a SHAP impact plot
plot.shap.summary(shap_scores_std)
```

References

- http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
- https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27
- https://blog.datascienceheroes.com/how-to-interpret-shap-values-in-r/