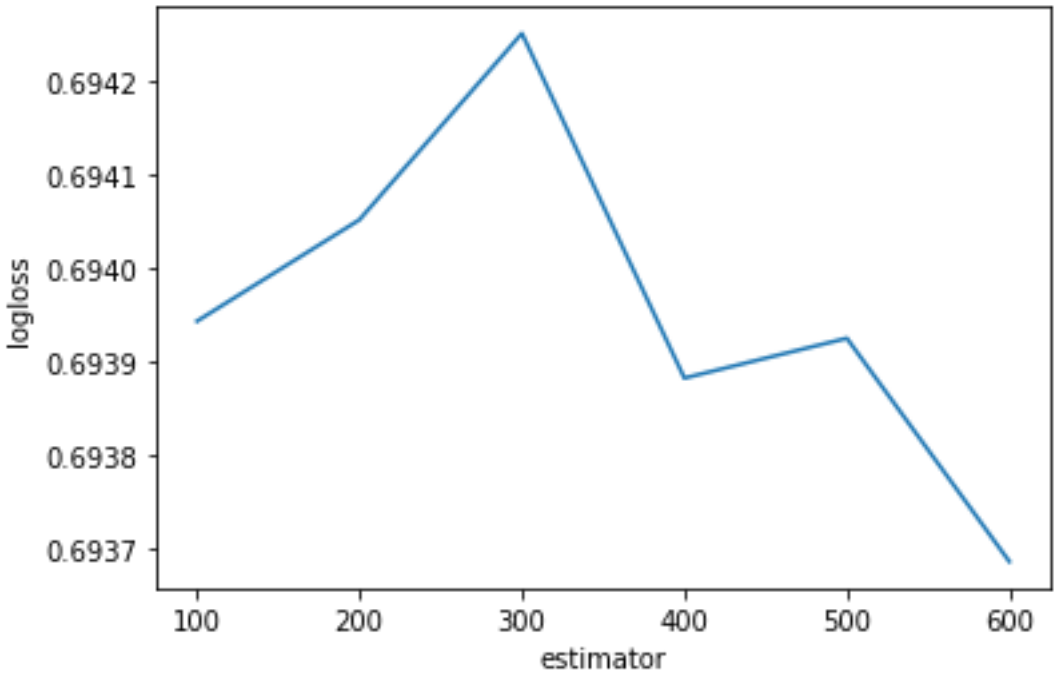
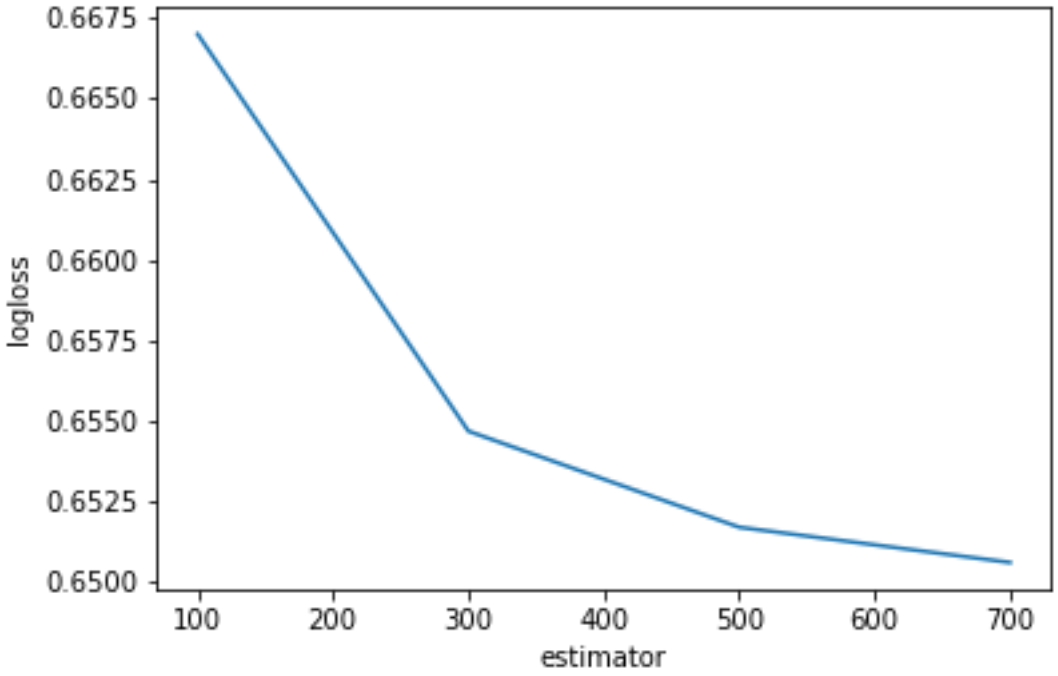
Github: https://github.com/DekaiLin/cmpt459project/tree/master/Milestone3
Random Forest: Dekai Lin, Gradient Boost: Zherui Shao, Naïve Bayes: Luowen Zhu

1. Random Forest, Gradient Boost and Naïve Bayes.
2. We first use ['bathrooms','bedrooms','latitude','longitude','price'] as our features because they are already there and we do not need to do calculation or transformation to get them.
   In Question 5, we add ['year','month','day','hour','minute'] and TF vector of the ['features'] as features for improvement. The ['year','month','day','hour','minute'] is derived from the ['created'] attribute.
3. We use "scores = cross_val_score(treeModel,X, y, cv=10,scoring = 'neg_log_loss')" to perfom 10-fold cross-validation. After that, we get 10 scores and average the scores to get the estimated performance about the classifier.
4. First version (['bathrooms','bedrooms','latitude','longitude','price']) performance.
   Random Forest logloss: 0.70034 (validation on training dataset), 0.71116 (test dataset).
   Gradient Boost logloss: 0.65717 (validation), 0.67689 (test).
   NB logloss (only 100 iteration): 0.81315 (validation), 0.88510 (test).
   Performance: Gradient Boost > Random Forest > NB
   The reason of NB has the worst performance is because NB classifier has the assumption that attribute values are conditionally independent given class label. However, in our selected features, 'bathrooms', 'bedrooms' and 'price' have some dependence on each other. It is common that more 'bedrooms' will have more 'bathrooms' also 'price' will be higher because more people can stay at the same time. So the conditionally independent assumption doesn't hold and the performance is bad.
   The Random Forest classifier performance is similar to Decision tree in Milestone 2.
   The Gradient Boost achieves the best performance. Compared with Random Forest which is creating an independent collection of estimators, Gradient Boost is designing the estimators work together. It will increase the weight of incorrect labeled points and train a new estimator based on the samples weight. So each subsequent tree concentrates on the example missed previously.
5. a. Tune Parameters:
   For the Random Forest and Gradient Boosting, we try with different number of the n_estimators, max_depth and min_samples_leaf. We also try different learning_rate of Gradient Boosting classifier.
   For the NB, it don't have parameter need to tune because the priors can calculated from training data distribution.





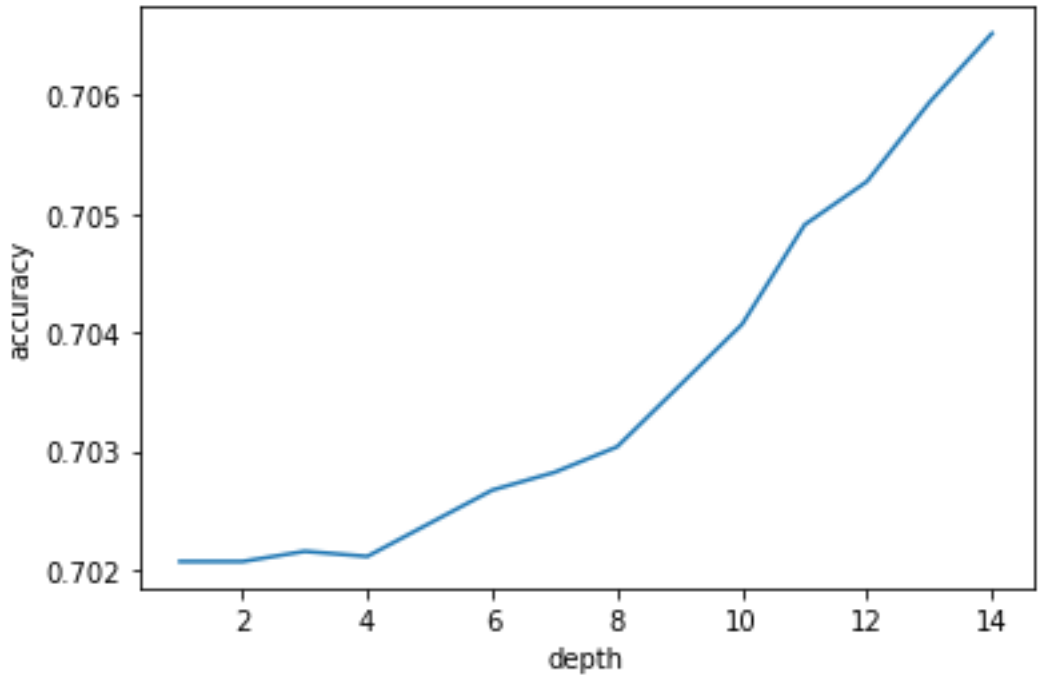
b. Use more features:

we add ['year','month','day','hour','minute'] into our training dataset. We derived these attributes from ['created'] attribute.

c. Add the unstructured attribute:

At this part, we try to use ['features'] as our new features. Firstly, we join the features in the ['features'] into a single string. So we can treat them as a short document, then we use CountVectorizer to create Term Frequency vector for each training object. Lastly, we hstack the new TF vector into the old attributes to create new training objects.

6. We compare the logloss on the training dataset and the test dataset, if the test dataset logloss is much bigger, it has overfitting. (For example, we have tried Random Forest with depth = 5 and depth = 10 at the second improvement. The logloss of depth = 5 is (0.67712 vs 0.69358) and for depth = 10 is (0.59077 vs 1.0254). So the difference of depth = 10 is too big, it is overfitting.). So, before we actually train the model, we try the cross validation on different parameters first to make sure we do not overfitting the model.
7. Random Forest:
   Validation dataset: 0.69819 → 0.69638 (a. tune parameter) → 0.67712 (b. use more features) → 0.58395 (c. add the unstructured data)
   Test dataset: 0.70964 → 0.70499 → 0.69065→ 0.67160
   Gradient Boosting:
   Validation dataset: 0.65717 → 0.60055 → 0.56266 → 0.52216
   Test dataset: 0.67689 → 0.67742 → 0.63959 → 0.59758
   NB:
   Validation dataset: 0.81315 → None parameter tuning → 0.83428 → 0.84212
   Test dataset: 0.88510 → None parameter tuning → 0.90537 → 0.86621
   What we did in the modification step are "Algorithm Tuning" and "Feature Engineering". Because the machine learning algorithms are driven by parameters, so these parameters majorly influence the outcome of learning process. In the tuning part, we tried to find out the optimal parameters by iterate some possible options and choose the ones have the best performance.
   In the Feature engineering part, we extract more information from existing data as our new features. These features may have a higher ability to explain the variance in the training data thus giving improved model accuracy. For example, we extract ['year', 'month', 'day' ···] from the ['created'] attribute. Here the ['created'] date may not have direct correlation with the ['interest_level'], but if we look at the year, month or day ··· it may have a higher correlation. So we unleash the hidden relationship of a data set in this step.
8. We use classification accuracy as our evaluation metric on the Random Forest.



It's not like the decision tree we trained in milestone 2. The accuracy metric and log loss metric is agreed with each other this time. So the accuracy metric and log loss metric in this classifier will tell us the same best parameter. (Accuracy on training dataset with depth = 15 on the last improvement is 0.70291.)

However, what we learned from the last milestone is that accuracy and logloss are measuring 2 different things so we can not directly compare this two value which makes no sense.

9. In this milestone, we try the NB classifier which is easy to use but have the bad performance and ensemble classifiers (Random Forest and Gradient Boosting) which needs more time to train but have a better performance. Compare with the milestone 2 classifiers performance, we get this rank: Gradient Boosting > Logistic Regression > Random Forest > Decision tree > SVM > NB.

   Gradient Boosting and Logistic Regression achieve the top log loss because both of them have the ability that use mistake of current model to improve the next model.

   Random Forest performance better than Decision tree is because it can reduce the effect of variance on a single decision tree.

   The SVM needs tons of time to train so we set the iteration only 100 in milestone2 which is so early for a SVM classifier converge. Nevertheless, the SVM classifier performance is still better than NB because NB needs an assumption that attributes are conditional independent given class label. But on the other hand, NB training time is much less than SVM because the algorithm itself is very simple to run.

10. The best logloss score we got is 0.59758 (the csv is also in the Github).