CS 205 Final Report

Teia Noel

June 6, 2019

1 Background

Here, we build machine learning models that leverage data from the National Health and Nutrition Examination Survey in attempt to classify individuals who have ever been told they had cancer and those who have not.

2 Methods

2.1 Feature Selection

Manual Feature Selection

Initially, I manually selected features based on my prior knowledge, online sources, and intuition regarding what might cause cancer. While my selections were backed by some reasoning, I tried to keep an open mind and not be too stringent. 80 features were selected in this initial step.

In the first category, demographics, I selected gender, age, and race because it is known that cancer affects categories of these demographics differently. For example, the rate of new cancer instances is higher in men than women, and highest in white and black individuals amongst all races. Moreover, there is a trend of increasing rate of new cancer instances with increasing age [1]. Additionally, I selected annual household income, and ratio of family income to poverty. These demographics were backed by health disparities known to occur amongst various income brackets. Lower socioeconomic status is associated with higher morbidity [2]. Specifically, lower-income families tend to have sparser access to healthy food options, safe living conditions, healthcare, and recreational facilities, weakening their general health condition [2]. The number of people in a household was an extension of this based on the intuition that a larger household would be more difficult to manage in terms of affording healthcare costs and high-quality, nutritious food, and maintaining a safe living environment. There were a number of features from the demographics section strategically left out from the model. Although education is considered an indicator of health [2], the educational features provided here seemed insufficient. For subjects aged 6-19, the feature merely stated the education level of each subject at the time of evaluation without differentiating those who lacked education. While it is expected that there exist health disparities amongst subjects aged 20+ lacking higher education, there was ample missing data for this feature. Country of birth was left out with the assumption that it would be correlated with race. Serving active duty, citizenship status, length of time in US, marital status, pregnancy status, language, total number of people in the family, and

number of people of different age brackets living in the household were all suspected to not be of much use predicting ever having had cancer because they are largely unrelated to health.

The second category of data I looked at was examination metrics. It is known that obesity is a cause of various forms of cancer [3]. Therefore, I mainly looked at metrics related to an unhealthy diet and lifestyle. For instance, I expected high 60-second heart rate, 60-second pulse, systolic blood pressure, diastolic blood pressure, weight, and BMI, and irregular pulse to distinguish unhealthy or obese individuals with relatively healthy individuals. For measures with multiple readings, including systolic blood pressure and diastolic blood pressure, I selected the first reading because I expected subsequent readings to be highly correlated. I ignored examination data related to oral health because this is not expected to be an indicator of a wide array of cancers, and while some metrics here might target mouth cancer, this is not amongst the more common cancers.

The third category of features I selected from was laboratory data. Here, I selected many measures of substances known to be carcinogenic, including cadmium in blood (μmol/L), chromium in blood (nmol/L), acrylamide in blood (pmol/G Hb), and iodine in urine (ng/mL) [4, 5, 6]. Additionally I selected substances, manganese in blood (μmol/L) lead in blood (μmol/dL), mercury in blood (μmol/L), and selenium in blood (μg/L); while there isn't strong evidence classifying these substances as carcinogenic, I assumed that high exposure to these compounds indicate constant exposure to an unhealthy living or working environment, a demographic position that would place someone at higher risk of health disparities. Because there were often multiple measures of each substance in blood and similar measures in urine, I arbitrarily chose one, assuming they would be highly correlated with each other. Additionally, I selected measures for high density cholesterol (mg/dL) and total cholesterol (mg/dL) because, once again, high values for either of these measures are likely to distinguish those with unhealthy lifestyles and diets with those who are healthy, a general causal factor of various forms of cancer. The HIV 1/2 combo test was chosen because HIV negatively affects immunity, and a hindered immune system is less likely to detect and fight off cancer cells [7]. Tests for hepatitis B core and surface antibodies were chosen because the disease is associated with lung cancer, one of the most common forms of cancer [7]. The oral HPV test results were chosen because HPV is known to be a cause of various forms of vaginal and anal cancers [8]. Chlamydia and herpes tests were chosen because negative results could be associated with irresponsible sexual behavior or poor sexual hygiene, leading to various forms of vaginal cancer. I chose urine flow rate (mL/min) because low urinary flow rate is a symptom of bladder cancer, another common form of cancer [9]. It is important to note that many of the features selected here were intended to sift out individuals who recently had or had cancer at the time of the evaluation. Those individuals with a recent diagnosis were more likely to have results positively correlated with the diagnosis.

Lastly, I looked into questionnaire data. Here, I sampled a few questionnaire questions from relevant categories. Firstly, I selected "general health condition", expecting a general evaluation of poor health to correspond with the general classification of ever having had cancer. I looked into questions regarding smoking behavior because smoking is known to be causal of various forms of cancer, including bladder, lung, cervical, and mouth/throat cancer [10]. This is largely due to the fact that smoking weakens the immune system, which would otherwise effectively fight off cancer cells. Specifically, I sought after a question that would distinguish heavy smokers from nonsmokers, and found the questions asking whether or not an individual smoked at least 100 cigarettes in her life, and the average number of cigarettes smoked per day in the last 30 days to be fitting. With similar intuition, I selected a question likely to distinguish heavy drinkers from casual drinkers, namely the question asking for the average number of drinks each individual had per day for the previous 30 days. Heavy drinking is associated with mouth, larynx, liver, throat, esophagus, and breast cancer [11]. Furthermore, questions asking if participants ever were told by a doctor that they were overweight, had high blood pressure, or had high cholesterol, were chosen as indicators of poor health and obesity. Likewise, number of frozen meals or pizza in a typical week and ever

participating in vigorous activities in a typical week was chosen to distinguish poor health and obesity from good health. The question regarding whether an individual was covered by health insurance was selected assuming that those who weren't lacked the sufficient access to healthcare to take preventative measures against diseases like cancer. The remaining questionnaire features were selected to seek out individuals who have or had specific forms of cancer. For instance, the number of times someone has had a sunburn in the last year, the number of minutes spent outdoors on a workday, and adverse skin reactions after a period of non-exposure are all positive indicators of ever having had skin cancer, and having both ovaries removed can be a result of ever having had ovarian cancer. Lastly, ever taking estrogen/progesterone pills, hormone therapy that is associated with breast cancer, one of the most common forms of cancer [12]. Questionnaire data that was ignored were those that asked questions likely to be correlated with other features more reliably measured. As examples, health/lifestyle-related questions more realistically covered by examination and laboratory data (e.g., measures of weight and BMI are better measurements than self-reported weight). Housing characteristics included number of rooms in an individual's home, and whether a home was bought or rented; the useful information here is how this correlates with income, which in turn positively correlates with general health quality. But this was item was many times measured more reliably with income and exam and laboratory tests. On the other hand, drug use was ignored because there seemed to be low subject compliance with answering these questions (many missing data points).

I ignored dietary data because this data largely consisted of dietary habits of subjects in recent days preceding the evaluation. Responses were thereby unlikely to be indicative of long-term eating/lifestyle habits, which were captured better by other questionnaire features.

L1-Based Feature Selection

Furthermore, L1-based feature selection was utilized on the manually selected features for a more stringent and automated selection layer. L1-based feature selection works by adding to the cost function of the logistic regression model a penalty of the form, $\lambda \sum_{i=1}^{n} |\beta_i|$, where n is the number of features and β_i for $i = 1, ..., n$ are the coefficients of each feature. For this reason, L1 regularization forces relatively smaller coefficients to zero in order to minimize the cost function, resulting in a sparse number of features. Furthermore, if two features are highly correlated, there is a chance that one of the two features will be arbitrarily chosen to remain in the model, while the other is eliminated because amongst the two options, there is no drastic difference in the cost. The L1-based feature selection technique simply built a L1-regularized logistic regression model, and returned features for which their corresponding coefficients were greater than 0. For this reason, this feature selection procedure was intended to eliminate features highly correlated with another feature, something that not always easy to gauge intuitively. After L1-based feature selection, 9 features were eliminated, and 71 remained.

Tree-Based Feature Selection

As a second automated feature selection technique, tree-based feature selection was chosen. This feature selection technique is based on feature importances, defined in python by the mean decrease in gini. Initially, the threshold for feature importances was selected to be the average feature importance, set by default. I opted for another means to select the threshold-I generated a random feature by drawing from a standard normal distribution and defined the threshold as its feature importance. This is done because feature importances are unreliable; they are affected by the range and continuity of the data [13]. Therefore, feature importances below the feature importance of a random feature with no structure is likely to be junk.

2.2 Preprocessing and Feature Engineering

Preprocessing techniques were partitioned into those used for real-valued features and those used for categorical features. Real-valued features were those with numerical values that were direct measures of the features at hand. Categorical features were those with numerical values encoding categories relevant to the features at hand. For categorical features with no ordering, I used the given one-hot encoder, "preproc_onehot," whereby each category was converted into its own feature column with a value of 1 for every individual falling into that category, and 0 otherwise.

There were a few exceptions to using "preproc_onehot." The customized preprocessing function "preproc_Herpes" was used to process the oral HPV and herpes type 1 and 2 tests. The values for all of these features were similar; 1 meant a positive result, 2 meant a negative result for all three features, while 3 meant that the individual was not evaluated for the HPV oral test, and indeterminate for the herpes tests. Because a small proportion of individuals had a value of 3 for all three tests, "preproc_Herpes" replaced 3 with the negative result, assuming most individuals would not be positive for these diseases. Because values encoded categories that did not have an inherent ordering, all values underwent the same one-hot encoding protocol described above.

"preproc_SkinReact" was used to process the categorical feature "Skin reaction to sun after non-exposure." Here, values 1-5 encoded various reactions to the sun after a period of non-exposure, but there wasn't a clear ordering (i.e., the severity of each reaction was somewhat subjective). The values 6, 77, and 99 referred to "Other," "Refused," and "Don't know," respectively. Because there were a total of 9 responses for values 77 and 99, in "preproc_SkinReact," these values were assigned the value 6 because of its neutrality. Lastly, the values underwent one-hot encoding for its final feature processing step.

Lastly, "preproc_Smoked100times" was used to process the features "Smoked at least 100 cigarettes in life," "Covered by health insurance," "Vigorous recreational activities," "Doctor ever said you were overweight," "Ever told you had high blood pressure," "Doctor told you-high cholesterol level," "Had both ovaries removed," "Used estrogen/progestin combo pills." For all of these features, the value 1 encoded "Yes," the value 2 encoded "No," and values 7 and 9 indicated "Refused" and "Don't know," respectively. Thus, "preproc_Smoked100times" set values 7 and 9 to 2, assuming that the average person would be in the negative category for all of the features processed with this function. Again, the values were processed using the one-hot encoder.

On the other hand, for categorical features assigned discrete values with ordinal meaning (e.g., increasing counting numbers assigned to increasing ranges of values), I simply used a variation of the "preproc_real" function, the "preproc_median" function, which maintains the assigned values and imputes the median instead of mean for missing values. I reasoned to impute the median instead of mean because it is less likely to be perturbed by outliers. Furthermore, like "preproc_real", if a cutoff value is specified, it will set all values above the cutoff to nan and impute the median for those values as well. Lastly, it will standardize the data.

There was one exception to using the "preproc_median" function for this purpose. Household annual income was a categorical feature that assigned values 1 through 10 to increasing income brackets. However, there was no bracket assigned to 11, and 12 encoded "$20,000 and Over", a redundant category since the value 5 referred to an income bracket with a minimum value of $20,000, and the value 10 encoded an income bracket with a maximum value of $74,999. Similarly, the value 13, corresponding to "Under $20,000" was redundant because the value 5 referred to a minimum income of $20,000, and the value 1 corresponded with a minimum income of 0. Moreover, increasing income brackets continued beyond the value 13. For instance, the values 14 and 15 referred to the income brackets "$75,000 to $99,999" and "$100,000 and over", respectively. For this reason,

the values were non-ordinal, and required a unique preprocessing function. In the function, "preproc_AnnualIncome", values in the feature column that are 12 or 13 are assigned nan, and values that are 14 and 15 are replaced with 11 and 12, respectively. The function also deals with a specified cutoff value, where values beyond the cutoff are assigned nan as well. In the function call to process the household annual income feature, 77 was the cutoff because the values 77 and 99 referred to responses, "Refused" and "Don't know." Lastly, "preproc_AnnualIncome" imputes the median for missing values, and standardizes all values.

For the same reason, I used the "preproc_median" function to preprocess real-valued features. There was one exception to using "preproc_median" for real-valued features. For instance, the questionnaire feature encoding the number of pizzas or frozen meals subjects ate in the past 30 days was preprocessed with the "preproc_FrozenPizza" function. This feature included real numbered values between 0 and 180, as well as 7777 encoding the response "Refused" and 9999 referring to the response "Don't know." Because each of these values encoded responses for one individual each, the function replaces 7777 and 9999 with nan. Missing values are then imputed with the median and all values are standardized.

2.3 Predictive Modeling

A number of classification models were chosen, making classification decisions within diverse feature spaces. For every classification model I chose, I ran it on all of the features initially manually selected, those selected with L1-based feature selection, and those selected with tree-based feature selection.

Linear Classifiers

The first set of models I built were linear models: regularized logistic regression, including lasso, ridge, and elastic net regularization, and support vector classifiers [14, 15]. Initially, I built the models with generic versions of each of the classifiers, largely with default parameters. Logistic regression with ridge regularization was run with the parameters max_iter=200 and solver='sag', and all other parameters set to default. Logistic regression with elastic net regularization was run with the parameters max_iter=500 and solver='saga', and all other parameters set to default. Logistic regression with lasso regularization was run with max_iter=200 and solver='liblinear', and all other parameters set to default. For each of these models, hyperparameter tuning was utilized, which searches through numerous combinations of parameters in order to select those that achieve the highest accuracy. For this purpose I used the functions LogisticRegressionCV and GridSearchCV in scikit learn for python [16, 17]. LogisticRegressionCV chose the regularizing coefficient for lasso, ridge, and elastic net across five cross validations that maximized the models' accuracy measures, while GridSearchCV chose optimal parameters across all combinations of specified parameters and five cross validations for the support vector classifier (for the gamma values 1, 0.1, and 0.001, and the kernels rbf and linear).

Non-Linear Classifiers

The second set of models were non-linear: random forest classifiers and k nearest neighbor classifiers [18, 19]. Initially, the classifiers were built from a few specified parameters, but largely default parameters determined by scikit learn. The random forest classifier was built with the parameters n_jobs=-1 and n_estimators=100, and all other parameters set to default. The KNN classifier was built with the parameter n_neighbors=3 and all other parameters set to default. Again, hyperparameter tuning was run here using GridSearchCV with five cross validations in order to hone in on parameters that achieved the highest accuracy. I used grid search to find the optimal parameters for the

random forest classifier amongst the "n_estimators" 50, 100, and 200, and "max_leaf_nodes" 10, 50, and 100. For k nearest neighbors I searched for the optimal parameters amongst the "n_neighbors" 3, 5, and 11, the "metrics" manhattan and euclidean, and the "weights" uniform and distance. Only odd numbers for the n_neighbors parameter were chosen so that the algorithm never ran into ties [20].

"Mixed" Classifiers

The classifiers chosen here leveraged the power of multiple machine learning algorithms into a single model.

Using the strongest-performing classifiers from largely differing feature spaces, I built a voting classifier that decided on the class of an individual based on the majority vote of five high-performing classifiers [21]. The motivation behind the voting classifier was the law of large numbers; the majority amongst a multitude of classifications is likely to reach a better consensus than any one classifier alone. The classifiers I chose for voting were one logistic regression classifier, a support vector classifier, a random forest classifier, and a k nearest neighbors classifier. For every voting classifier I built, all participating models were those run on the same set of features (all, L1-selected, or tree-selected), and for every category of classifiers I chose those with the highest accuracy amongst those hyperparameter-tuned and those not tuned. A diverse set of classification models was chosen to participate in the voting in order to ensure that they are largely independent, creating uncorrelated errors.

Additionally, adaboosting was used on the strongest classifier amongst each group of features selected (manual, L1-based, and tree-based) to focus on training data the classifier underfitted [22].

With the assumption that the features might have had complicated interactions, I built a linear SVM on top of a neural network. The way it worked was as follows: the neural network makes a non-linear transformation of the features such that it learns a representation of the data that is effective for classification with a linear SVM; i.e., it tries to find a higher-level feature space that is linearly separable. The model is trained by passing the hinge-loss error from the SVM to the neural network, thus training it using back propagation (Figure 1).

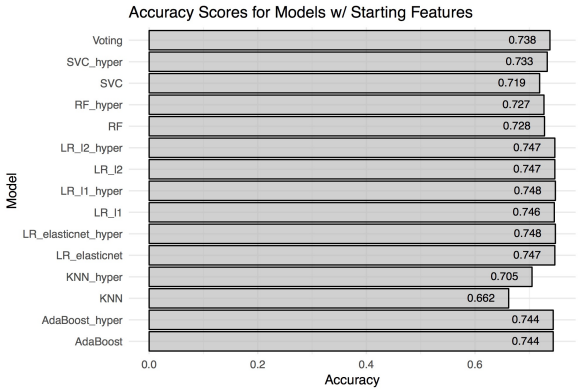
nn with svm

```
In [28]: def clamp(preds):
             for i in range(len(preds)):
                 preds[i][0] = max(min(preds[i][0],1),0)
                 preds[i][1] = max(min(preds[i][1],1),0)
             return preds

In [39]: model_svm = Sequential()

         model_svm.add(Dense(200,input_dim = n_fe)) #n_fe (#of features) as input dimension. 200 neurons
         model_svm.add(Activation('relu'))# Rectified Linear Unit as non-linearity
         model_svm.add(Dropout(0.5))# drops some neurons during each training epoch for regularization
         model_svm.add(Dense(50)) # 50 neuron layer
         model_svm.add(Activation('relu'))

         model_svm.add(Dropout(0.5))

         #model.add(Dense(1, kernel_regularizer=l2(0.01)))
         #model.add(Activation('linear'))

         model_svm.add(Dense(2, kernel_regularizer=l2(0.1))) #two outputs (probability for each class). I
         model_svm.add(Activation('linear')) #Linear activation for output layer


         model_svm.compile(loss='categorical_hinge', #hingle loss + linear activation for output layer +
                           optimizer='adam',
                           metrics=['accuracy'])

In [40]: model_svm.fit(features_trn, ttargets_trn, epochs = 50, batch_size = 500 )
```

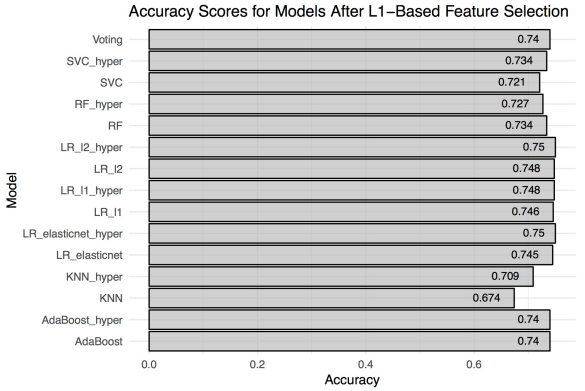
Figure 1

3 Results

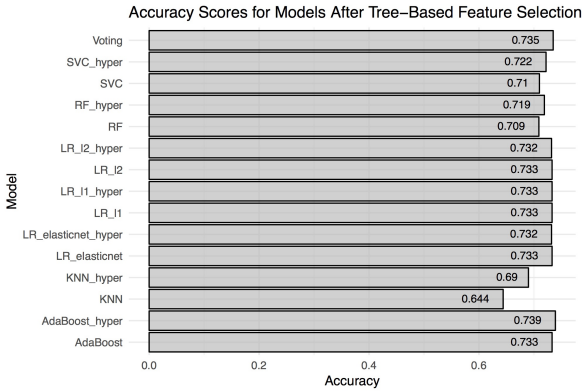
3.1 Classification Accuracy Measures



(a)



(b)



(c)

Figure 2: Accuracy scores for models built from (a) all manually-selected features, (b) l1-selected features, and (c) tree-selected features

Model Label	Description
Voting	Voting classifier
SVC_hyper	Support vector classifier with hyperparameter tuning
SVC	Support vector classifier with gamma='auto' and all other parameters set to default
RF_hyper	Random forest classifier with hyperparameter tuning
RF	Random forest classifier with n_jobs=-1, n_estimators=100, and all other parameters set to default
LR_l2_hyper	Logistic regression model with ridge regularization and hyperparameter tuning
LR_l2	Logistic regression model with ridge regularization, max_iter=200, solver='sag', and all other parameters set to default
LR_l1_hyper	Logistic regression model with lasso regularization and hyperparameter tuning
LR_l1	Logistic regression model with lasso regularization, max_iter=200, solver='liblinear', and all other parameters set to default
LR_elasticnet_hyper	Logistic regression with elastic net regularization and hyperparameter tuning
LR_elasticnet	Logistic regression with elastic net regularization, max_iter=500, and solver="saga"
KNN_hyper	K nearest neighbors classifier with hyperparameter tuning
KNN	K nearest neighbors classifier with n_neighbors=3 and all other parameters set to default
AdaBoost_hyper	AdaBoosting of logistic regression with hyperparameter tuning
AdaBoost	AdaBoosting of logistic regression with n_estimators=100 and all other parameters set to default

Table 1: Figure 1 model labels and their descriptions

For the models run on all initially-selected features, all logistic regression models performed the best and comparably with accuracies between 0.747 and 0.748. The runner-ups were the AdaBoosting models, both default and hyperparameter-tuned, resulting in accuracies of 0.744. On the other hand, the K nearest neighbor classifiers, with and without hyperparameter tuning, performed the worst with accuracies of 0.705 and 0.662, respectively, largely because KNN does not work well with high-dimensional data. The overall trends here were that hyperparameter tuning often improved the accuracy of each model, and that linear models performed better than non-linear models. This is illustrated by the highest accuracies achieved by AdaBoosting and logistic regression models, as well as the increase in accuracy between the generic SVC model and the SVC model with hyperparameter tuning, wherein the linear model was optimal (Figure 2a). Lastly, the neural network/SVC model achieved an accuracy of 0.7255 on all features. This model was not run with feature selection with hopes of finding complex interactions amongst as many features as possible. In any case, it likely didn't achieve the highest accuracy here because it seems that linear models performed better, and since this model was built after seeing the results of all other models, I didn't bother to run it with further feature selections.

For the models run on L1-selected features, again, all logistic regression models achieved the highest accuracies, with logistic regression with ridge regularization and hyperparameter tuning and logistic regression with hyperparameter tuning and elastic net regularization achieving the absolute highest accuracies of 0.75, which was higher than the highest accuracies of the raw model. Again, the AdaBoosting models came in second with accuracies at 0.74, a worse accuracy than AdaBoosting with the original features. Here, the voting classifier achieved the same accuracy as AdaBoosting. The KNN models performed the worst with accuracies of 0.709 and 0.674 for the hyperparameter tuned model and generic model, respectively, albeit higher accuracies than the previous models. Once again, this result is due to the fact that KNN does not perform well with high dimensional data, and even after L1-based feature selection, we maintained 71 features. Following the trends of the models run on all of the features, the linear models achieved higher accuracies than the nonlinear models, and hyperparameter-tuned models tended to achieve higher accuracy than their respective generic models (Figure 2b).

For the models run on tree-selected features, we see similar trends as before, with logistic regression models performing best, AdaBoosting and voting classifiers performing second-best, and KNN models performing worst. General trends that once again appeared were the linear models outperforming non-linear models and hyperparameter tuning increasing the accuracy of each of the models. However, none of the models performed better than their corresponding models with the original features and L1-selected features (Figure 2c).

3.2 Feature Coefficients of the Most Accurate Models

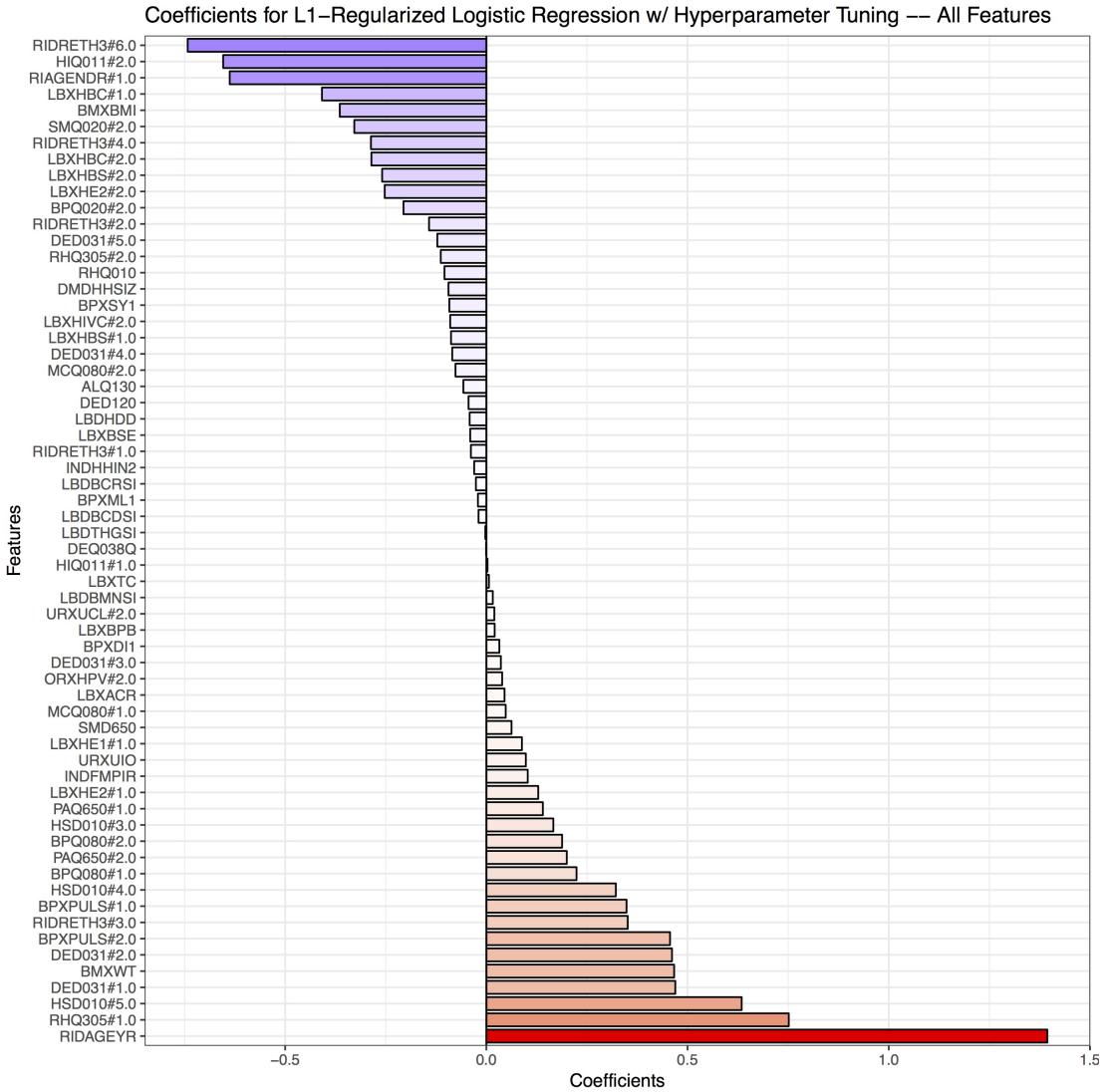


Figure 3

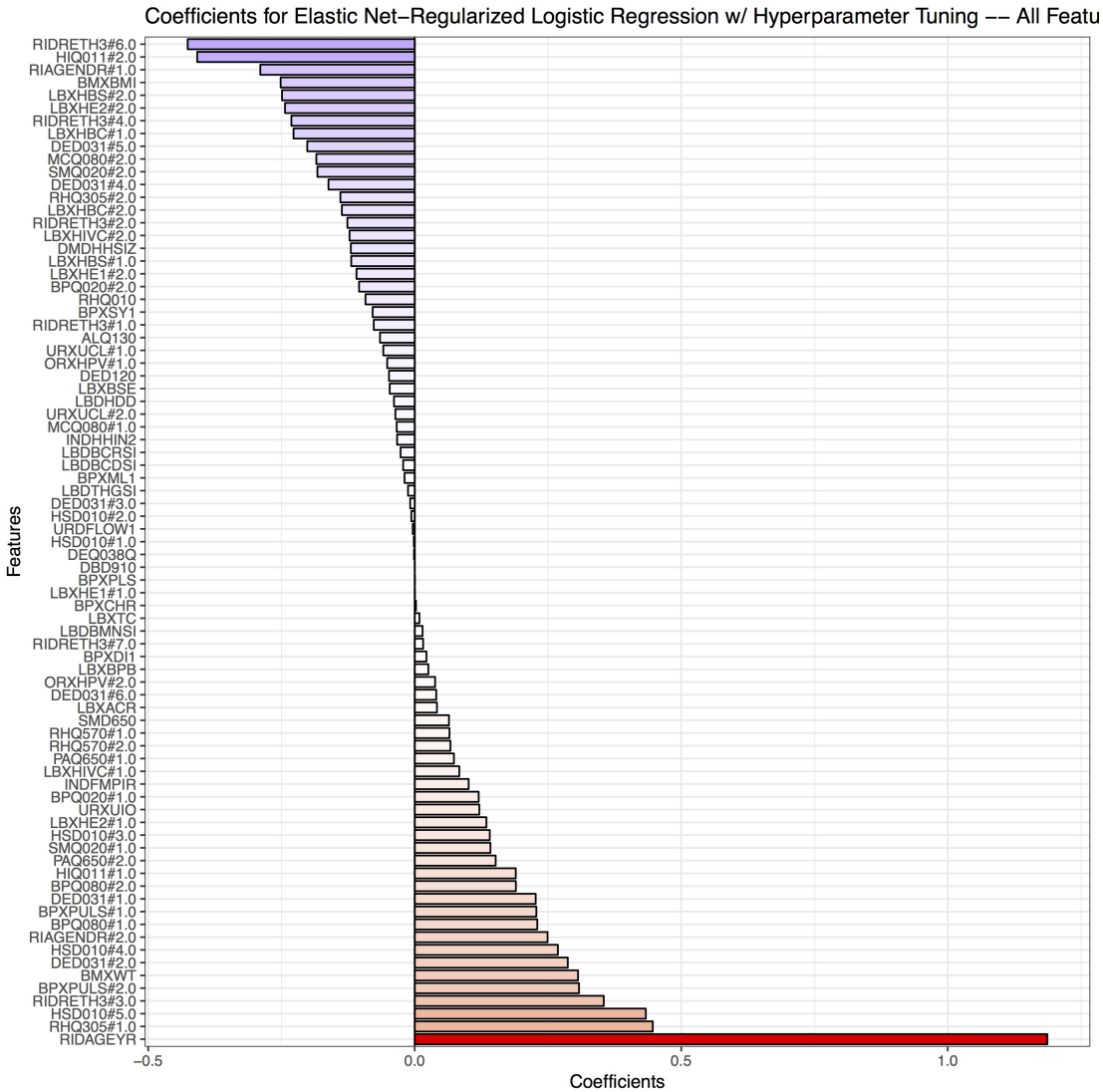

Coefficients for Elastic Net−Regularized Logistic Regression w/ Hyperparameter Tuning −− All Featu
Features
RIDRETH3#6.0
HIQ011#2.0
RIAGENDR#1.0
BMXBMI
LBXHBS#2.0
LBXHE2#2.0
RIDRETH3#4.0
LBXHBC#1.0
DED031#5.0
MCQ080#2.0
SMQ020#2.0
DED031#4.0
RHQ305#2.0
LBXHBC#2.0
RIDRETH3#2.0
LBXHIVC#2.0
DMDHHSIZ
LBXHBS#1.0
LBXHE1#2.0
BPQ020#2.0
RHQ010
BPXSY1
RIDRETH3#1.0
ALQ130
URXUCL#1.0
ORXHPV#1.0
DED120
LBXBSE
LBDHDD
URXUCL#2.0
MCQ080#1.0
INDHHIN2
LBDBCRSI
LBDBCDSI
BPXML1
LBDTHGSI
DED031#3.0
HSD010#2.0
URDFLOW1
HSD010#1.0
DEQ038Q
DBD910
BPXPLS
LBXHE1#1.0
BPXCHR
LBXTC
LBDBMNSI
RIDRETH3#7.0
BPXDI1
LBXBPB
ORXHPV#2.0
DED031#6.0
LBXACR
SMD650
RHQ570#1.0
RHQ570#2.0
PAQ650#1.0
LBXHIVC#1.0
INDFMPIR
BPQ020#1.0
URXUIO
LBXHE2#1.0
HSD010#3.0
SMQ020#1.0
PAQ650#2.0
HIQ011#1.0
BPQ080#2.0
DED031#1.0
BPXPULS#1.0
BPQ080#1.0
RIAGENDR#2.0
HSD010#4.0
DED031#2.0
BMXWT
BPXPULS#2.0
RIDRETH3#3.0
HSD010#5.0
RHQ305#1.0
RIDAGEYR
−0.5
0.0
0.5
1.0
Coefficients

Figure 4

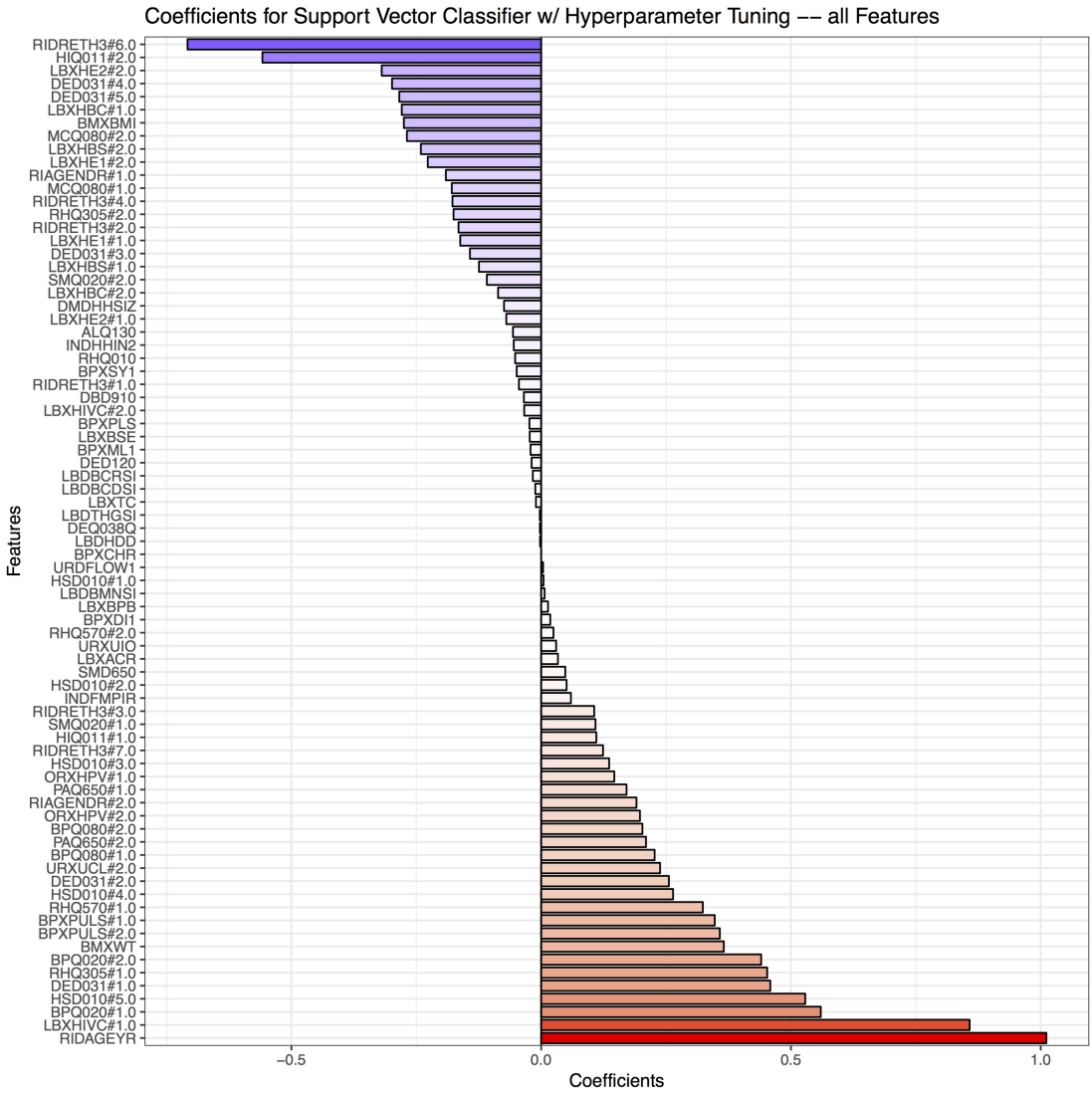

Coefficients for Support Vector Classifier w/ Hyperparameter Tuning –– all Features
Features
RIDRETH3#6.0
HIQ011#2.0
LBXHE2#2.0
DED031#4.0
DED031#5.0
LBXHBC#1.0
BMXBMI
MCQ080#2.0
LBXHBS#2.0
LBXHE1#2.0
RIAGENDR#1.0
MCQ080#1.0
RIDRETH3#4.0
RHQ305#2.0
RIDRETH3#2.0
LBXHE1#1.0
DED031#3.0
LBXHBS#1.0
SMQ020#2.0
LBXHBC#2.0
DMDHHSIZ
LBXHE2#1.0
ALQ130
INDHHIN2
RHQ010
BPXSY1
RIDRETH3#1.0
DBD910
LBXHIVC#2.0
BPXPLS
LBXBSE
BPXML1
DED120
LBDBCRSI
LBDBCDSI
LBXTC
LBDTHGSI
DEQ038Q
LBDHDD
BPXCHR
URDFLOW1
HSD010#1.0
LBDBMNSI
LBXBPB
BPXDI1
RHQ570#2.0
URXUIO
LBXACR
SMD650
HSD010#2.0
INDFMPIR
RIDRETH3#3.0
SMQ020#1.0
HIQ011#1.0
RIDRETH3#7.0
HSD010#3.0
ORXHPV#1.0
PAQ650#1.0
RIAGENDR#2.0
ORXHPV#2.0
BPQ080#2.0
PAQ650#2.0
BPQ080#1.0
URXUCL#2.0
DED031#2.0
HSD010#4.0
RHQ570#1.0
BPXPULS#1.0
BPXPULS#2.0
BMXWT
BPQ020#2.0
RHQ305#1.0
DED031#1.0
HSD010#5.0
BPQ020#1.0
LBXHIVC#1.0
RIDAGEYR
−0.5
0.0
0.5
1.0
Coefficients

Figure 5

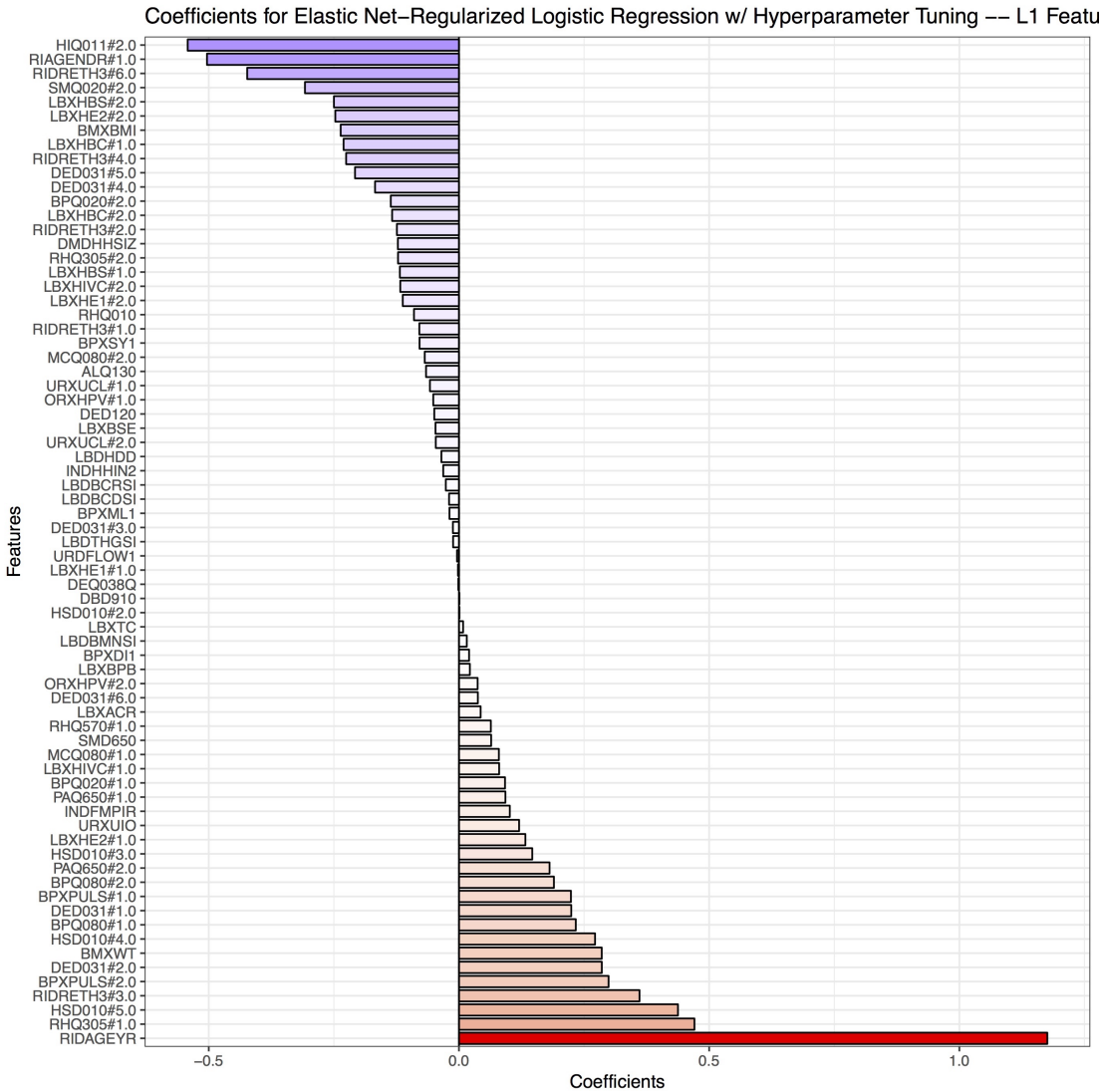

Coefficients for Elastic Net−Regularized Logistic Regression w/ Hyperparameter Tuning −− L1 Featu
HIQ011#2.0
RIAGENDR#1.0
RIDRETH3#6.0
SMQ020#2.0
LBXHBS#2.0
LBXHE2#2.0
BMXBMI
LBXHBC#1.0
RIDRETH3#4.0
DED031#5.0
DED031#4.0
BPQ020#2.0
LBXHBC#2.0
RIDRETH3#2.0
DMDHHSIZ
RHQ305#2.0
LBXHBS#1.0
LBXHIVC#2.0
LBXHE1#2.0
RHQ010
RIDRETH3#1.0
BPXSY1
MCQ080#2.0
ALQ130
URXUCL#1.0
ORXHPV#1.0
DED120
LBXBSE
URXUCL#2.0
LBDHDD
INDHHIN2
LBDBCRSI
LBDBCDSI
BPXML1
DED031#3.0
LBDTHGSI
URDFLOW1
LBXHE1#1.0
DEQ038Q
DBD910
HSD010#2.0
LBXTC
LBDBMNSI
BPXDI1
LBXBPB
ORXHPV#2.0
DED031#6.0
LBXACR
RHQ570#1.0
SMD650
MCQ080#1.0
LBXHIVC#1.0
BPQ020#1.0
PAQ650#1.0
INDFMPIR
URXUIO
LBXHE2#1.0
HSD010#3.0
PAQ650#2.0
BPQ080#2.0
BPXPULS#1.0
DED031#1.0
BPQ080#1.0
HSD010#4.0
BMXWT
DED031#2.0
BPXPULS#2.0
RIDRETH3#3.0
HSD010#5.0
RHQ305#1.0
RIDAGEYR
Features
−0.5
0.0
0.5
1.0
Coefficients

Figure 6

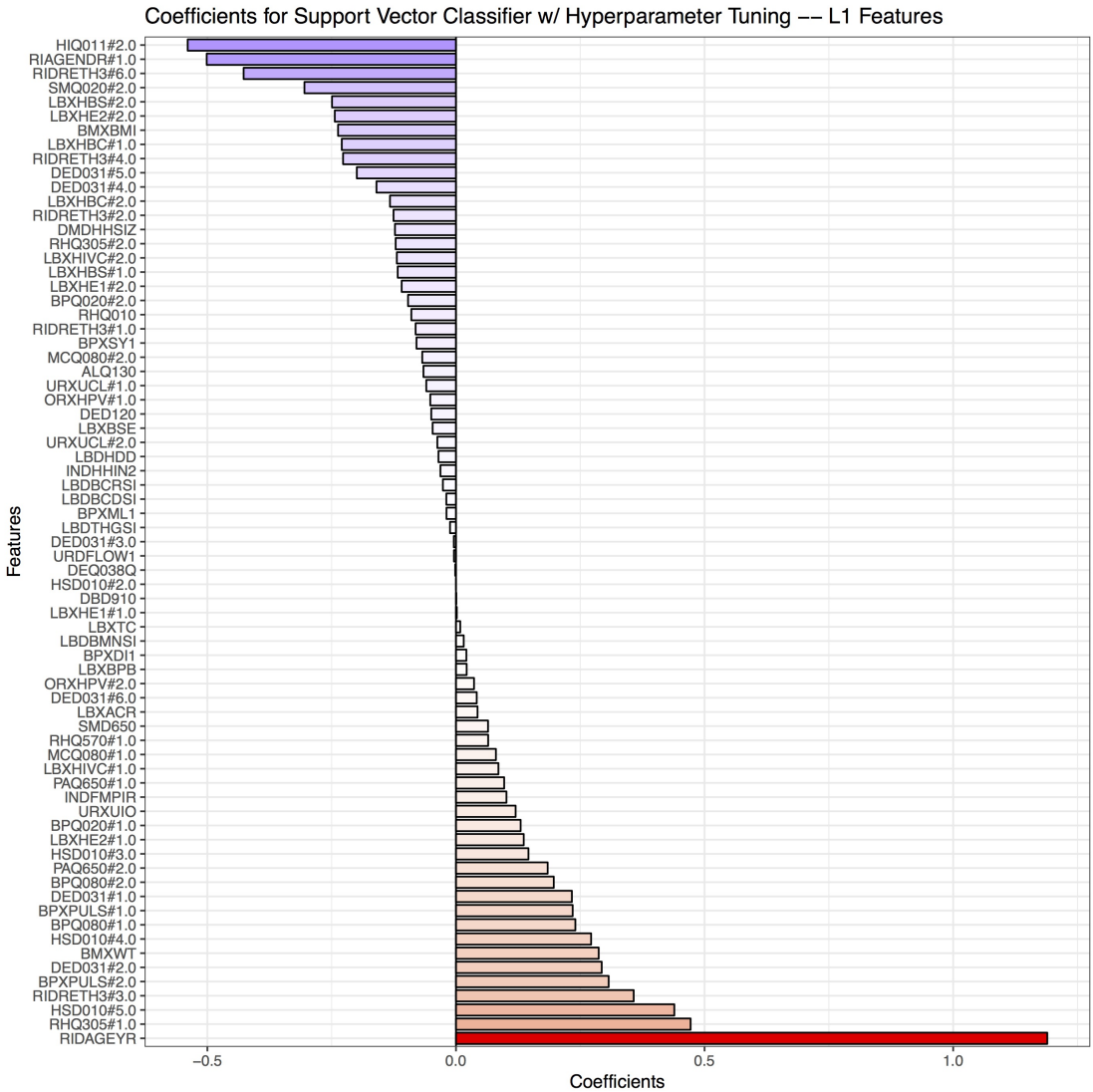

Coefficients for Support Vector Classifier w/ Hyperparameter Tuning –– L1 Features
HIQ011#2.0
RIAGENDR#1.0
RIDRETH3#6.0
SMQ020#2.0
LBXHBS#2.0
LBXHE2#2.0
BMXBMI
LBXHBC#1.0
RIDRETH3#4.0
DED031#5.0
DED031#4.0
LBXHBC#2.0
RIDRETH3#2.0
DMDHHSIZ
RHQ305#2.0
LBXHIVC#2.0
LBXHBS#1.0
LBXHE1#2.0
BPQ020#2.0
RHQ010
RIDRETH3#1.0
BPXSY1
MCQ080#2.0
ALQ130
URXUCL#1.0
ORXHPV#1.0
DED120
LBXBSE
URXUCL#2.0
LBDHDD
INDHHIN2
LBDBCRSI
LBDBCDSI
BPXML1
LBDTHGSI
DED031#3.0
URDFLOW1
DEQ038Q
HSD010#2.0
DBD910
LBXHE1#1.0
LBXTC
LBDBMNSI
BPXDI1
LBXBPB
ORXHPV#2.0
DED031#6.0
LBXACR
SMD650
RHQ570#1.0
MCQ080#1.0
LBXHIVC#1.0
PAQ650#1.0
INDFMPIR
URXUIO
BPQ020#1.0
LBXHE2#1.0
HSD010#3.0
PAQ650#2.0
BPQ080#2.0
DED031#1.0
BPXPULS#1.0
BPQ080#1.0
HSD010#4.0
BMXWT
DED031#2.0
BPXPULS#2.0
RIDRETH3#3.0
HSD010#5.0
RHQ305#1.0
RIDAGEYR
Features
–0.5
0.0
0.5
1.0
Coefficients

Figure 7

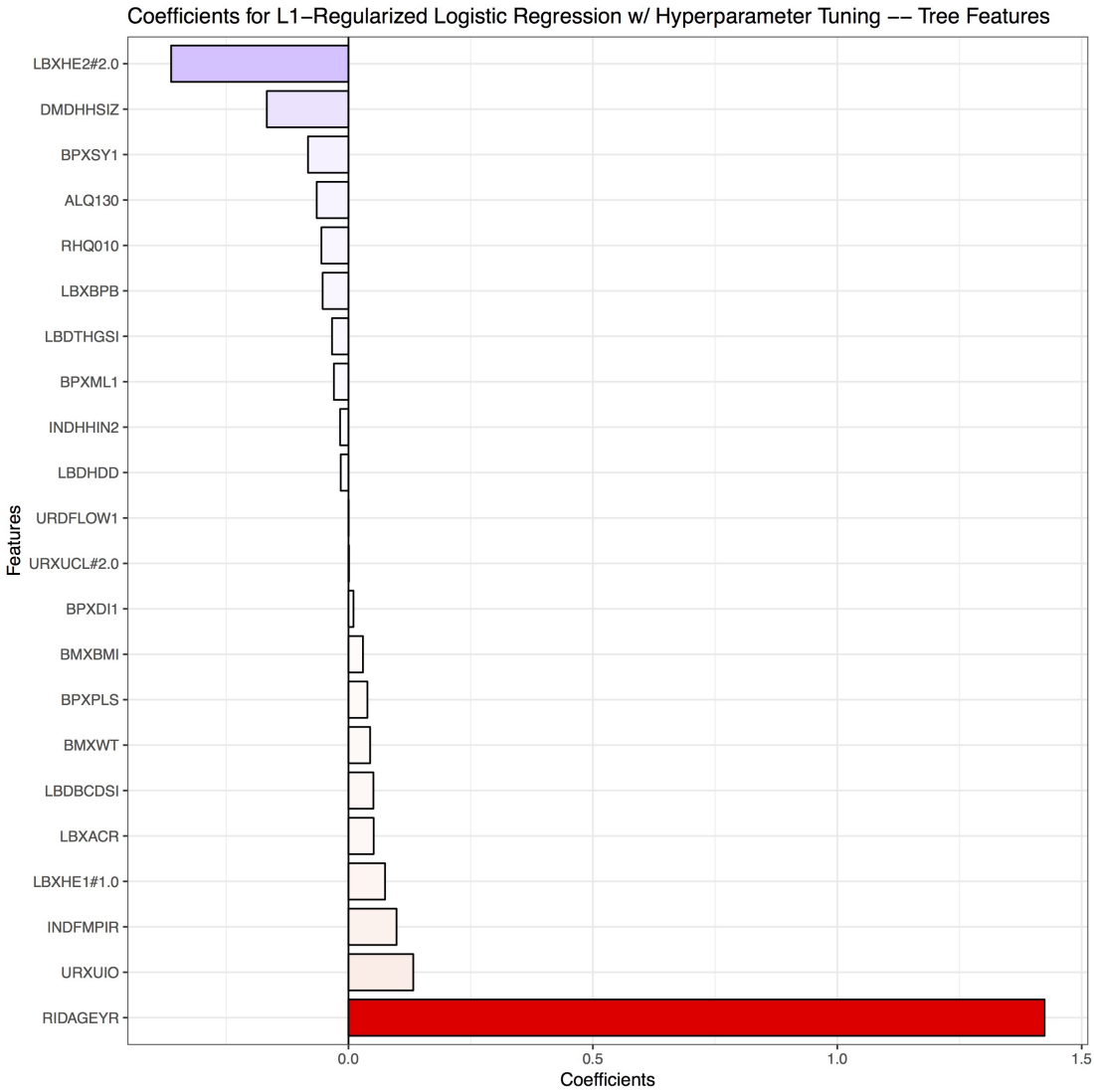


Figure 8

Non-zero coefficients for all linear models with the highest accuracies were evaluated. Out of the models built upon manually-selected features, logistic regression with lasso regularization and hyperparameter tuning, logistic regression with elastic net regularization and hyperparameter tuning, and the support vector classifier with hyperparameter tuning all performed best (FIgure 2a). Out of the models built upon L1-selected features, logistic regression with elastic net regularization and hyperparameter tuning and the support vector classifier with hyperparameter tuning performed best (Figure 2b). Out of the models built upon tree-based features, logistic regression with lasso regularization and hyperparameter tuning (Figure 2c) performed best.

Out of all the models, the most predictive feature, positively correlating with age, was age (Figures 3-8). This makes sense because, as mentioned before, age is known to explain a lot of the variance associated with cancer instances [1]. And when we're faced with the task of predicting generally ever having had cancer, it makes sense that a category as general and widely varying as age would be the most predictive feature. Although this doesn't amount to anything novel, this did account for variance in the data, allowing us to hone in on other predictive features with non-zero coefficients

that could account for the remaining variance. However, no other feature with positive coefficients compared with the magnitude of the coefficient for cancer across all high-performing models.

However, the most highly negative coefficients varied across high-performing models. For logistic regression with lasso regularization and hyperparameter tuning on all features, logistic regression with elastic net regularization and hyperparameter tuning on all features, the logistic regression model with elastic net regularization and hyperparameter tuning on L1-based features, and the support vector classifier with hyperparameter tuning on the L1-based features, the three largest negative coefficients corresponded with the ethnicity, "Other Race-Including Multi-Racial," not being covered by health insurance, and being male (Figures 3, 4, 6, and 7). While being multi-racial and not having health insurance were unexpected negative predictors of cancer, it is known that cancer affects men more than females. Moreover, negative coefficients were found for testing negative for the herpes simplex virus type 2 in the logistic regression model with lasso regularization and hyperparameter tuning on tree-based features and the support vector classifier with hyperparameter tuning for all features (Figures 5 and 8). Although not thoroughly researched and found to be associated with a specific cancer, this again makes sense because poor sexual hygiene can be associated with a range of vaginal cancers.

References

[1] "Rate of Cancers: All Types of Cancer, United States, 2015." https://gis.cdc.gov/Cancer/USCS/DataViz.html. Accessed: 2019-05-28.

[2] N. E. Adler and K. Newman, "Socioeconomic disparities in health: pathways and policies," *Health affairs*, vol. 21, no. 2, pp. 60–76, 2002.

[3] "Obesity and Cancer." https://www.cancer.gov/about-cancer/causes-prevention/risk/obesity/obesity-fact-sheet. Accessed: 2019-06-01.

[4] "Cancer-Causing Substances." https://www.cancer.gov/about-cancer/causes-prevention/risk/substances. Accessed: 2019-06-02.

[5] "Acrylamide and Cancer Risk." https://www.cancer.gov/about-cancer/causes-prevention/risk/diet/acrylamide-fact-sheet. Accessed: 2019-06-02.

[6] S. Kargar, S. M. Shiryazdi, S. R. Atashi, H. Neamatzadeh, and M. Kamali, "Urinary iodine concentrations in cancer patients," *Asian Pacific journal of cancer prevention: APJCP*, vol. 18, no. 3, p. 819, 2017.

[7] "Immunosuppression." https://www.cancer.gov/about-cancer/causes-prevention/risk/immunosuppression. Accessed: 2019-06-01.

[8] "HPV and Cancer." https://www.cancer.gov/about-cancer/causes-prevention/risk/infectious-agents/hpv-and-cancer#cancers-caused. Accessed: 2019-06-01.

[9] "Common Cancer Types." https://www.cancer.gov/types/common-cancers. Accessed: 2019-06-01.

[10] "Smoking and Cancer." https://www.cdc.gov/tobacco/campaign/tips/diseases/cancer.html. Accessed: 2019-06-01.

[11] "Alcohol." https://www.cancer.gov/about-cancer/causes-prevention/risk/alcohol. Accessed: 2019-06-01.

[12] "Menopausal Hormone Therapy and Cancer." https://explained.ai/rf-importance/index.html. Accessed: 2019-06-02.

[13] "Beware Default Random Forest Importances." https://www.cancer.gov/about-cancer/causes-prevention/risk/diet/acrylamide-fact-sheet. Accessed: 2019-06-02.

[14] H.-F. Yu, F.-L. Huang, and C.-J. Lin, "Dual coordinate descent methods for logistic regression and maximum entropy models," *Machine Learning*, vol. 85, no. 1-2, pp. 41–75, 2011.

[15] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[16] "Logistic Regression CV." https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html. Accessed: 2019-06-06.

[17] "Grid Search." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed: 2019-06-06.

[18] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19] "K Nearest Neighbors." https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed: 2019-06-06.

[20] "k-Neighbors Classifier with GridSearchCV Basics." https://medium.com/@erikgreenj/k-neighbors-classifier-with-gridsearchcv-basics-3c445ddeb657. Accessed: 2019-06-06.

[21] "Voting Classifier." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html. Accessed: 2019-06-06.

[22] "AdaBoost Classifier." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html. Accessed: 2019-06-06.