IEOR 142 Final Project Report: Shelter Animal Adoption Predictor

Team 1: Alexandra Novales, Anita Tam, Ian Glynn, Majd Muna, Vasanth Madhavan

Problem Motivation, Objectives, and Impact

One of the biggest challenges of animal shelters is the allocation of ~6.5 million rescued animals. Unfortunately, ~1.5 million rescued animals are euthanized every year. Our goal is to develop a model to predict which animals have a higher probability of being adopted at shelters. In determining this, efforts may be more properly allocated to increase the number of adoptions and decrease the number of euthanizations at shelters.

We aim to gain insight on which pet attributes contribute to the likelihood of adoption and their significance. In turn, the model outcome may help shelters identify which animals have a smaller chance of being adopted and dedicate greater effort to find these animals a home. Additionally, this predictor model may aid in providing an estimation on which and how many animals will be adopted, which can help with more effective allocation of resources (ie. space, equipment, etc) and give revenue projections.

Data Source and Preparation

We use public shelter animal intake and outcome records provided by Sonoma County, CA to train our predictor. This data includes features of the animal such as their name, age, sex, neuter status, species/breed, color, size, health condition, outcome type (ie. adoption, transfer, etc), and more. We begin with about 22,130 records and 24 columns in the original dataset. Most of the features recorded in the dataset are categorical, which were encoded into binary columns. Many columns had missing data, which were omitted or encoded as an 'Other' or 'Not Applicable' binary column and used as the reference level incorporated into the intercept. We choose to aggregate the 'Age' column into 4 groups to reflect 'baby', 'young', 'mid-aged', or 'old' groups. The original dataset included outcome records of animals transferred to other facilities and/or returned to their owners, which were not applicable to our adoption predictor model; these were omitted from the dataset. The final dataset comprised of about 8,520 records and 212 encoded features. A list of all the features are found in the appendix. Additionally, our dependent variable is 'Outcome Type' where

$1 = adopted\ or\ returned\ to\ owner$ and

$0 = passed\ away,\ disposed\ or\ euthanized.$

Lastly, depending on the model, we chose to look at both accuracy and the false negative rate of the model. We chose to evaluate accuracy, as it easily conveys model strength to non-technical audiences, such as those who work at shelters. False negatives are equally important, because the point of the model is to predict which pets will more likely be adopted so shelters can better allocate resources to prevent other animals from not being adopted. If we falsely predict an animal to not be adopted, they will most likely lose their chances of being adopted if moved elsewhere. We set different costs associated with true positives, true negatives,

false positives, and false negatives, with more details under '*Choosing cutoff threshold, p'* in the Appendix.

Analytical Models (What methods did you use? What were the results? How confident are you in your results?)
We built 5 models to test and compare the efficiency and performance of each as a shelter animal adoption predictor.

Logistic Regression
We build a logistic regression model in the form

$$P(Y = 1|X) = \frac{1}{1+exp(-f(X))},\ where\ f(X) = B_0 + B_1X_1 + \ldots + B_kX_k$$

where X_i is the i-th feature in our cleaned dataset, B_0 is the intercept, B_i is the coefficient to X_i.

We build an initial logit model with about 20 of the 212 encoded features in our cleaned data set using the statsmodel package in Python to easily obtain the significance and coefficients for each feature. We keep features with p-values under 0.15 to minimize loss and add in new features to the model in replacement of those omitted. We repeat this manual iteration through all 212 features.

Ideally, the model would be initially fed in all 212 variables and those with low significance would be removed one-by-one to observe its impact on the model. However, with so many potential features, we encounter a runtime error on statsmodel, hence the iterative introduction and omittance of features to/from the model. We often encounter singular matrix errors, which were resolved through elimination of certain features before their incorporation into the model. Thus, the logit model may not be the most optimal in this case.

After feature selection, we are narrowed down to 42 final features. Some top predictive features for this model are the type of animal, whether the animal was fertile, their health condition, their age, certain dog breeds and certain colors. More details are in the appendix.

The logistic regression model performs very well on the test set and shows great improvement compared to the baseline model in accuracy (69% vs. 95%). We used a threshold of p = 0.13 to minimize expected penalty, which yields a much lower FNR than using p = 0.5. We bootstrapped 1000 test sets and achieved a 95.5% accuracy and 0.28% FNR. The performance metrics of the 42-features logit model are highly similar to that of the 212-features model, which shows that our feature selection was successful. An in-depth comparison and explanation for threshold choice is found under 'Logistic Regression' in the appendix.

Linear Discriminant Analysis (LDA)
LDA involves $P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\Sigma_{l=0,1}\pi_l f_l(x)}$, and the discriminant function

$_k = x.\frac{\mu_k}{(\sigma)^2} - \frac{\mu_k}{2(\sigma)^2} + log(\pi_k)$, where $f_k(x)$ follows the normal distribution, π_k is the

marginal probability, $\boldsymbol{\mu}_k$ is the class mean, and σ is the common standard deviation for class k. The algorithm uses maximum likelihood estimation to estimate the aforementioned parameters.

Building an LDA model, we used the sci-kit learn package in Python, fitting all the features as independent variables and Outcome Type as the dependent variable. The probability threshold used for the following was a default 0.5. Upon fitting the model and calculating performance metrics through the confusion matrix, the model was determined to be 95.66% accurate on the test set. Additionally, TPR was a high 99.55% and FNR was a low 0.45%, whereas FPR and TNR were 12.92% and 87.08% respectively. When using p=0.13 to minimize expected penalty (appendix), FNR falls to a lower 0.06% at the cost of accuracy also decreasing to 94.13% and FPR increasing to 18.70%. This reflects respectable performance with the LDA model which can help with adoption outcome predictions, as it performed highly with predicting true positives and false negatives. This gives us high confidence in the LDA model and its corresponding results.

Classification and Regression Trees (CART)

We build a CART model to solve our classification problem by building a classification tree. We choose $(j, s)$to minimize:

$$\Sigma_{i:x_i \epsilon R_1(j,s)}(y_i - \widehat{y_1}(j,s))^2 \; + \; \Sigma_{i:x_i \epsilon R_2(j,s)}(y_i - \widehat{y_2}(j,s))^2$$

where $R_1(j,s) = \{X: X_j < s\}$, $R_2(j,s) = \{X: X_j \geq s\}$. This process of splitting is done recursively as learned in class. The CART model will split each subsequent region based on which split decreases the total impurity the most as defined in the lecture notes. After initializing the CART model, we performed 10-fold cross validation in order to find the best complexity parameter (cp) with the highest accuracy. The best cp was 0.0005, with the highest accuracy of 95.27%. Once implemented and used on the test set, our final tree has 75 nodes and an accuracy of 96.55%. Additionally, the model has a false negative rate of 3.84%.

Similar to logistic regression, the CART model performs very well on the test set and shows great improvement compared to the baseline model in accuracy (69% vs. 97%). Due to the constraints of a CART model, instead of using a threshold of p = 0.13, we intuitively place more weight on FNR when looking at accuracy and FNR to compare which model does the best in bootstrapping. We bootstrapped 1000 test sets and achieved a similar accuracy of 96.56% and FNR of 1.64%.

Random Forest

When building our Random Forest model, we opted to create two models: one where we did not conduct any cross validation for the max number of features, and where we did. In order to create a Random Forests model, this would require the use of bagging, where we also set a parameter *m* which denotes the number of features that, when splitting each individual CART model, at each potential split, we only consider the said number of randomly selected features. When performing the bagging, we must also set a value *B* for how many different bootstrapped

training sets we'll generate (the number of estimates). From the bagging function, in order to obtain predictions, we must take the majority vote (or average the leafs node proportion values) via:

$$\hat{f}_{\text{bag}}(x) := \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x)$$

As such, for our Random Forest model with no correlation, in python, we set the maximum number of features to consider when looking for the best split to 5, the minimum number of samples required to be a leaf node to 5, and the number of trees in the forest (the number of estimators) to 500. These were the standard values introduced to us via the course labs. Having done so, given our data, our Random Forest model using no cross-validation yielded an accuracy of 92.14%, a true positive rate of 98.52%, a false positive rate of 21.96%, a true negative rate of 78.04%, and a false negative rate of 1.48%.

After having constructed this model, we moved on to create a second Random Forest model, but this time applying cross-validation with respect to the maximum number of features. We tested integer values between 1 and 120 and after having conducted our cross validation, we found that the best parameter value for the maximum number of features was 113 when using accuracy as our performance metric. To better visualize this result, we created a plot to show the relationship between the input value of max_features and the model's accuracy, which can be found in the appendix.

As such, we then created our new Random Forest model, this time utilizing the value of 113 for the maximum number of features (and keeping the rest of the input parameters the same) and this new model yielded and accuracy of 95.85%, a true positive rate of 98.52%, a false positive rate of 10.04%, a true negative rate of 89.96%, and a false negative rate of 1.48%.

Additionally, we also decided to create a bar graph that displays, for the cross-validated Random Forest model, which feature was the most important in determining whether or not an animal will end up being adopted. The graph can be found in the appendix, however, from what can be seen, features such as whether the animal's health condition, it's age, and gender were the most definitive features in determining this. The most important feature among them was if the animal's intake condition was deemed as being untreatable. In terms of our two separate models, while they both have the same low false negative rate of 1.48%, our cross-validated Random Forest model has a higher accuracy rate of 95.85%, resulting in it being the strongest of the two models, and independently, still being a strong model.

Gradient Boosting

We also created a Boosting Model using scikit learn's GradientBoosterClassifier. This technique was very effective out-of-bag using reasonable parameters. The model gave a test set accuracy of 95.36%. To further improve upon this, we attempted to use cross validation. We learned that attempting to cross validate a model with a large dataset and large grid of values is quite difficult.

Training this model took significantly more time than others, but we were eventually able to use cross validation to choose the n_estimators parameter. The new test accuracy is 95.36%. This model had a particularly high TPR at 98.69%.

Results

We set out to gain insights on what makes an animal likely to get adopted or euthanized. We created 5 different classification models that all had great test set performance. This is a result of careful feature engineering and parameter tuning. All models offered different benefits, and we can gain insights from them all. Gradient Boosting resulted in the highest test accuracy. Logistic Regression, CART and Random Forest gave information about what features are most important. We chose Logistic Regression and CART to further analyze, due to the interpretability and ability to easily incorporate custom loss. We bootstrapped the results of these two models to confirm both models had great performance.

Impact and Further Works

With the research, coding, and analysis undergone in this project, we built significant tools that predict the outcome of animals in shelter. Through these tools, we learned which pet attributes contribute more heavily to the likelihood of animal adoption. Because of the significance of these attributes, shelters can now better identify which animals have a smaller chance of being adopted. Shelters can consequently put more effort into finding these animals homes. Furthermore, estimating the quantity and outcome of shelter stay can allow shelters to better allocate their resources (space, equipment, etc.) which also helps with their respective financial projections.

Despite this, further research can be done to expand the scope of our analysis and strengthen our toolset. Our initial desired data, qualitative data such as personality traits of each respective animal, would go a long way. With natural language processing to analyze the data, we can find certain traits and characteristics that would influence adoption. This can theoretically improve the predictive capabilities of our tools to be even more accurate, since it focuses more on individual animals. Additional research would also include changing the model objective to predict if an animal is adopted in x amount of days, where we add another time factor into the model. The current state of our model may predict an animal gets adopted, even if it takes the animal months to get adopted. With this extra temporal layer, shelters can even better allocate their resources. Furthermore, we can test around with other models such as neural networks.

Appendix

Code Compilation Folder can be found here.
Data source: Animal Shelter Intake and Outcome from Sonoma County, CA found here.

Data Cleaning
Code for **data cleaning** can be found here.

Our original dataset had the following data about a rescued animal:

- **Name** - large variety
- **Type** - Dog, Cat, Other
- **Breed** - large variety
- **Color(s)** - large variety
- **Sex** - Female, Neutered, Spayed, Male, Unknown
- **Size** - 'MED', 'SMALL', 'KITTN', 'LARGE', 'TOY', 'PUPPY', nan, 'X-LRG'
- **Date of birth** - large variety
- **Impound Number** - large variety, not relevant information for our model
- **Kennel Number** - large variety, not relevant information for our model
- **Animal ID** - large variety, not relevant information for our model
- **Intake date** - date the animal entered the shelter, formatted month/day/year
- **Outcome Date** - date the animal left the shelter, formatted month/day/year
- **Days in Shelter** - discrete integer
- **Intake type** - 'STRAY', 'OWNER SURRENDER', 'CONFISCATE', 'QUARANTINE', 'ADOPTION RETURN', 'TRANSFER', 'OS APPT'
- **Intake subtype** - large variety
- **Outcome Type** - 'RETURN TO OWNER', 'ADOPTION', 'TRANSFER', 'EUTHANIZE', 'DIED', 'RTOS', nan, 'DISPOSAL', 'APPT', 'ESCAPED/STOLEN'
- **Outcome Subtype** - large variety
- **Intake Condition** - 'UNKNOWN', 'HEALTHY', 'TREATABLE/MANAGEABLE',
- 'TREATABLE/REHAB', 'UNTREATABLE'
- **Outcome Condition** - 'PENDING', 'HEALTHY', 'TREATABLE/MANAGEABLE', 'UNTREATABLE', nan, 'TREATABLE/REHAB', 'DEAD';
- **Intake jurisdiction** - 'SANTA ROSA', 'COUNTY', '*SONOMA', '*WINDSOR', 'OUT OF COUNTY', '*ROHNERT PARK', '*COTATI', '*PETALUMA', '*HEALDSBURG', '*SEBASTOPOL', '*CLOVERDALE', '*TRIBAL RESV', 'UNKNOWN'
- **Outcome jurisdiction** - 'SANTA ROSA', '*PETALUMA', 'COUNTY', nan, 'OUT OF COUNTY', '*SEBASTOPOL', '*ROHNERT PARK', '*COTATI', '*WINDSOR', '*HEALDSBURG', '*SONOMA', '*CLOVERDALE', 'UNKNOWN'
- **Outcome zip** - large variety

- **Location** - zip code and latitude, longitude coordinates
- **Count** - quantity of animal incoming; all values equal 1

We removed the columns Impound Number, Kennel Number, Animal ID, Outcome Date, Days in Shelter, Intake subtype, Outcome Subtype, Outcome Condition, Outcome jurisdiction, Outcome zip, Location, and Count as these columns are either information the shelter would not have when the animal is taken into the shelter or does not give any value in building our classifier.

We cleaned and prepared the remaining data columns:

- **Name** - 1 if name is common (if the name appears over 20 times in the dataset), 0 otherwise
- **Type** - Dog, Cat, Other (reference level)
- **Breed** - Breeds that appear more than 14 times
- **Color(s)** - Colors that appear more than 70 times. Mixed colors are listed as 'COLOR1/COLOR2' and were left in this format to signify mixed colored animals.
- **Sex** - Female, Neutered, Spayed, Male, Unknown (reference level)
- **Size** - 'MED', 'SMALL', 'KITTN', 'LARGE', 'TOY', 'PUPPY', 'X-LRG', nan (reference level)
- **Date of birth** - used to obtain an animal's age (in years) when entering the shelter
- **Intake date** - extract months and years and encoded them as binary, categorical columns
- **Intake type** - 'STRAY', 'OWNER SURRENDER', 'CONFISCATE', 'QUARANTINE', 'ADOPTION RETURN','TRANSFER', 'OS APPT' (reference level)
- **Intake Condition** - 'HEALTHY', 'TREATABLE, 'UNTREATABLE', 'UNKNOWN' (reference level)
- **Intake jurisdiction** - 'SANTA ROSA', '*SONOMA', '*WINDSOR', 'OUT', '*ROHNERT PARK', '*COTATI', '*PETALUMA', '*HEALDSBURG', '*SEBASTOPOL', '*CLOVERDALE', '*TRIBAL RESV', 'UNKNOWN' (reference level)
- **Outcome Type** - 1 if 'RETURN TO OWNER' or 'ADOPTION', 0 if 'EUTHANIZE', 'DIED', 'DISPOSAL'; else omitted

A complete list of all encoded features and descriptions can be found here.

Logistic Regression

Code for logistic regression modeling, feature selection, and test metrics can be found here.

Instructions on how to run the code:

1. Download folder linked above.
2. The file '1_LogitModel_FeatureSelection.ipynb' can be run on Python jupyter and shows the iterative feature selection process in building the logistic regression model.
3. The file '2_LogitModel_Metrics_p50.ipynb' can be run on Python jupyter and shows metrics such as accuracy, TPR, TNR, FPR, FNR, AUC, and ROC curves for the test set using the threshold p = 0.5.
4. The file '3_LogitModel_Metrics_p13.ipynb' can be run on Python jupyter and shows metrics such as accuracy, TPR, TNR, FPR, FNR, AUC, and ROC curves, and bootstrapping for the test set using the threshold p = 0.13.
5. The PDF files '1_LogitModel_FeatureSelection.pdf', '2_LogitModel_Metrics_p50.pdf', '3_LogitModel_Metrics_p13.pdf' are pdf versions of the 3 Python notebook mentioned above in steps 2-4.
6. All csv files are inputs to the Python notebooks for modeling, feature selection, and metric computations.

List of features after feature selection

Diagram C.1 - Final Features to Logistic Regression Model after Feature Selection

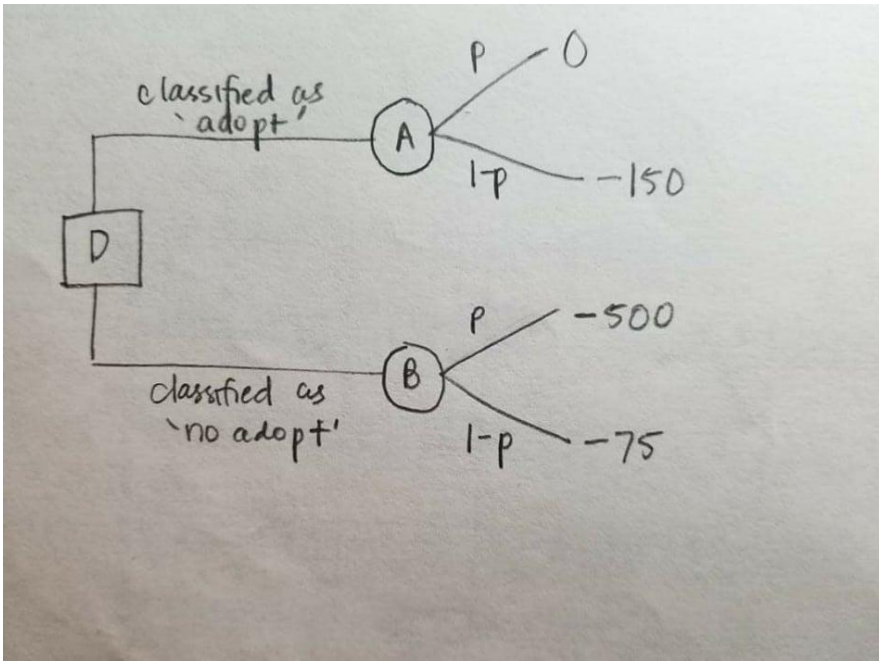
```
Optimization terminated successfully.
         Current function value: 0.088627
         Iterations 10
                          Logit Regression Results
==============================================================================
Dep. Variable:           Outcome Type   No. Observations:                 5963
Model:                          Logit   Df Residuals:                     5920
Method:                           MLE   Df Model:                           42
Date:                Mon, 06 Dec 2021   Pseudo R-squ.:                  0.8571
Time:                        18:42:49   Log-Likelihood:                -528.48
converged:                       True   LL-Null:                       -3699.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
==================================================================================================
                                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                             -0.8755      0.399     -2.194      0.028      -1.657      -0.094
Type_DOG                          -9.6872      0.732    -13.240      0.000     -11.121      -8.253
Type_CAT                          -7.4333      0.957     -7.764      0.000      -9.310      -5.557
Sex_Neutered                       5.9026      0.485     12.160      0.000       4.951       6.854
Sex_Spayed                         6.3973      0.499     12.816      0.000       5.419       7.376
Intake Type_OWNER SURRENDER       -1.0100      0.193     -5.246      0.000      -1.387      -0.633
Intake Type_QUARANTINE            -4.3721      0.560     -7.802      0.000      -5.470      -3.274
Intake Condition_HEALTHY           2.9570      0.298      9.928      0.000       2.373       3.541
Intake Condition_TREATABLE         1.2763      0.282      4.526      0.000       0.724       1.829
Intake Condition_UNTREATABLE      -5.2308      0.474    -11.032      0.000      -6.160      -4.302
Common_Name                        4.3585      0.438      9.951      0.000       3.500       5.217
Uncommon_Name                      4.4653      0.409     10.906      0.000       3.663       5.268
Intake Month_7                     0.7973      0.359      2.223      0.026       0.094       1.500
Intake Year_2021                   1.3618      0.448      3.043      0.002       0.485       2.239
Intake Year_2020                   1.1376      0.405      2.809      0.005       0.344       1.931
Intake Year_2018                   1.1048      0.318      3.471      0.001       0.481       1.729
Intake Year_2019                   1.6062      0.360      4.465      0.000       0.901       2.311
Intake Year_2014                   0.6329      0.262      2.413      0.016       0.119       1.147
Intake Year_2017                   0.8341      0.285      2.924      0.003       0.275       1.393
Intake Age 0-2                     2.5395      0.251     10.136      0.000       2.048       3.031
Intake Age 3-6                     0.6723      0.253      2.656      0.008       0.176       1.168
Intake Age 12-30                  -1.6219      0.356     -4.560      0.000      -2.319      -0.925
Breed_chihuahua sh                 1.1825      0.492      2.402      0.016       0.218       2.147
Breed_domestic sh                 -2.0355      0.763     -2.666      0.008      -3.532      -0.539
Breed_domestic lh                 -1.8040      0.864     -2.087      0.037      -3.498      -0.110
Breed_domestic mh                 -1.8650      0.846     -2.205      0.027      -3.523      -0.207
Breed_pit bull                    -2.2403      0.266     -8.436      0.000      -2.761      -1.720
Breed_rottweiler                  -1.2257      0.854     -1.436      0.151      -2.899       0.448
Breed_rabbit sh                   -8.2948      2.235     -3.711      0.000     -12.676      -3.914
Breed_catahoula                   -3.5114      1.170     -3.001      0.003      -5.805      -1.218
Breed_germ sh point               -3.8956      1.466     -2.657      0.008      -6.769      -1.022
Breed_alaskan husky               -2.5918      1.121     -2.313      0.021      -4.788      -0.395
Breed_dachshund                    2.5292      1.450      1.744      0.081      -0.313       5.371
Breed_maltese                      1.8683      0.984      1.899      0.058      -0.060       3.796
Breed_belg malinois                6.7019      1.367      4.903      0.000       4.023       9.381
Breed_amer bulldog                -2.0446      0.887     -2.305      0.021      -3.783      -0.306
Breed_pig                         -3.7041      2.625     -1.411      0.158      -8.848       1.440
Breed_rhod ridgeback              -2.0168      1.223     -1.649      0.099      -4.414       0.380
Breed_parakeet                    -2.7650      1.145     -2.415      0.016      -5.009      -0.521
Color_blue                        -1.1140      0.695     -1.602      0.109      -2.477       0.249
Color_gray                         1.1533      0.517      2.230      0.026       0.140       2.167
Color_org tabby/white             -1.5418      0.644     -2.392      0.017      -2.805      -0.279
Color_tan/black                   -2.0061      0.636     -3.155      0.002      -3.252      -0.760
==================================================================================================

Possibly complete quasi-separation: A fraction 0.15 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Most of our final features have p-value < 0.15 with a few exceptions. Variables with p-value > 0.15 were kept to optimize training accuracy.

Choosing cutoff threshold, p
We choose p = 0.13 through the following analysis:

- If a shelter correctly predicts an animal's adoption, they incur no loss
- If a shelter incorrectly predicts an animal's adoption, they incur losses associated with misallocation of space, resources, etc. We set this loss equal to 150.
- If a shelter correctly predicts that an animal will not be adopted, they incur losses associated with proper management or euthanization of the animal. We set this loss equal to 75.
- If a shelter incorrectly predicts that an animal will not be adopted, they incur losses associated with euthanization and a high cost of the incorrect disposal of life. This loss is difficult to quantify, and we choose a high loss of 500.

Diagram C.2 - Decision Tree



In Diagram C.2, *p* is the probability an animal will be adopted. The classification of animals (the decision) is made before we know whether the animal would've been adopted.

Obtaining p = 0.13:

$$LTP(p) + LFP(1-p) = LFN(p) + LTN(1-p)$$

$$0(p) - 150(1-p) = -500(p) - 75(1-p)$$

Solving the above equation gives p = 0.13.

Performance Metrics Comparison

Table C.3: Logistic Regression Test-set Metrics using p = 0.5

Model	Test Accuracy	TPR	FPR	TNR	FNR	AUC
Baseline	68.83%	100%	100%	0%	0%	0.5
Feature-selected (42 features) Model	96.13%	98.9%	9.9%	90.1%	1.1%	0.945
All-features (212 features) Model	96.21%	98.9%	9.8%	90.2%	1.1%	0.946

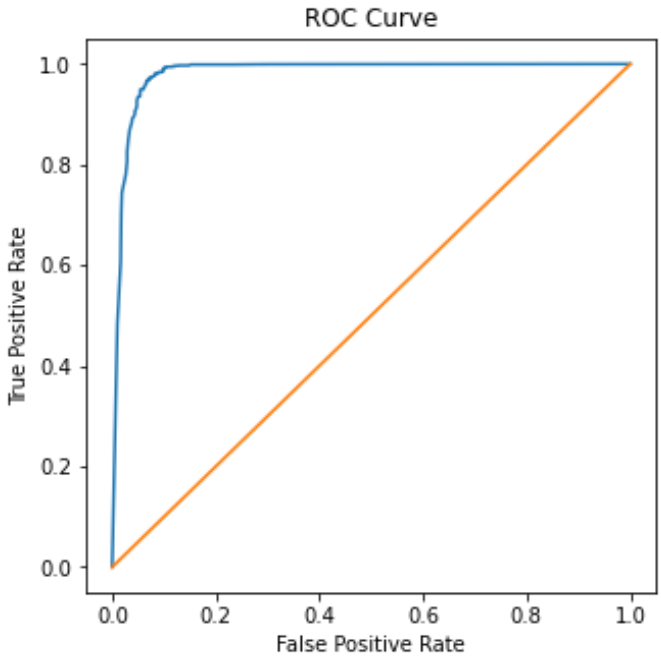
Table C.4: Logistic Regression Test-set Metrics using p = 0.13

Model	Test Accuracy	TPR	FPR	TNR	FNR	AUC
Baseline	68.83%	100%	100%	0%	0%	0.5
Feature-selected (42 features) Model	95.46%	99.7%	13.9%	86.1%	0.3%	0.929
All-features (212 features) Model	95.58%	99.8%	13.7%	86.32%	0.2%	0.930

Note: the 212-feature model was built using sci-kit learn, as opposed to the 42-feature model built on statsmodel.

As seen in Table C.2 and C.3, the performance of the metrics measured are highly similar between the feature-selected model and the all-features model. Using p = 0.5 as the threshold yields a slightly higher accuracy, but using p = 0.13 yields a much lower FNR, which is ideal to lower losses.

Graph C.5 - 42-features Logit Model ROC Curve for Test Set for p = 0.13



Orange line is the baseline ROC. Blue curve is the feature-selected model ROC, which closely resembles the all-features model ROC.

Because of the closeness in accuracy and other metrics, the ROC curve for p = 0.13 and p = 0.5 for both 42-features and 212-features model show high resemblance with each other.

CART

Code for CART modeling can be found here.

Instructions on how to run the code:

1. Download folder linked above.
2. The .ipynb file reflects the code, which can be run on a Python Jupyter notebook. It shows all modeling and testing.
3. The .csv file shows the dataset, which is read in the code.
4. The PDF file is a PDF version of the code for easier interpretation.

Table C.6: CART Test-set Metrics

Model	Test Accuracy	TPR	FPR	TNR	FNR	AUC
Baseline	68.83%	100%	100%	0%	0%	0.5
All-features (212 features)	95.56%	96.17%	3.29%	96.71%	3.83%	N/A

Linear Discriminant Analysis (LDA)

Code for LDA modeling, data used, and test metrics can be found here.

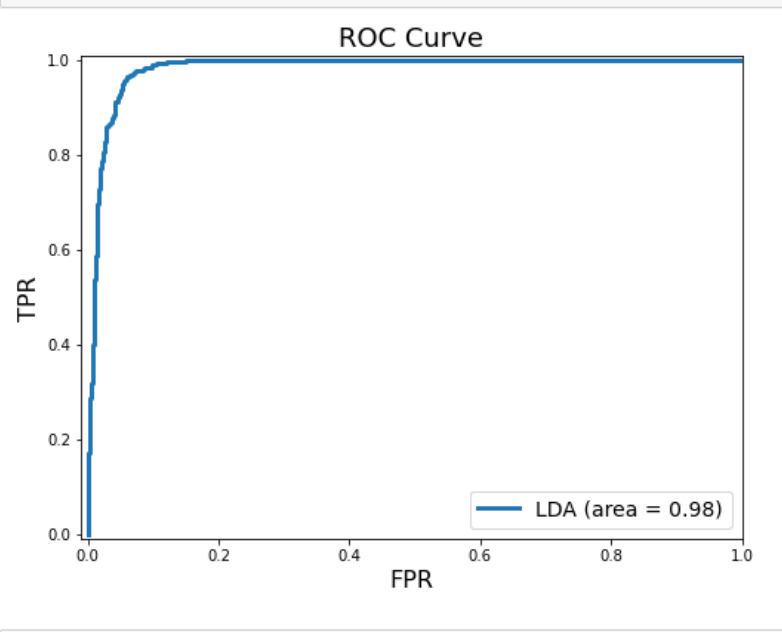
Instructions on how to run the code:

1. Download folder linked above.
2. The .ipynb file reflects the code, which can be run on a Python Jupyter notebook. It shows all modeling and testing.
3. The .csv file shows the dataset, which is read in the code.
4. The PDF file is a PDF version of the code for easier interpretation.

Table C.6: LDA Test-set Metrics

Model	Test Accuracy	TPR	FPR	TNR	FNR	AUC
Baseline	68.83%	100%	100%	0%	0%	0.5
All-features (212 features) Model with p=0.13	94.13%	99.94%	18.70%	81.30%	0.06%	0.98
All-features (212 features) Model with p=0.5	95.66%	99.55%	12.92%	87.08%	0.45%	0.98

Graph C.7: LDA Model ROC Curve on Test-Set



Random Forest Model

Code for the Random Forest modeling, data used, and test metrics can be found here.

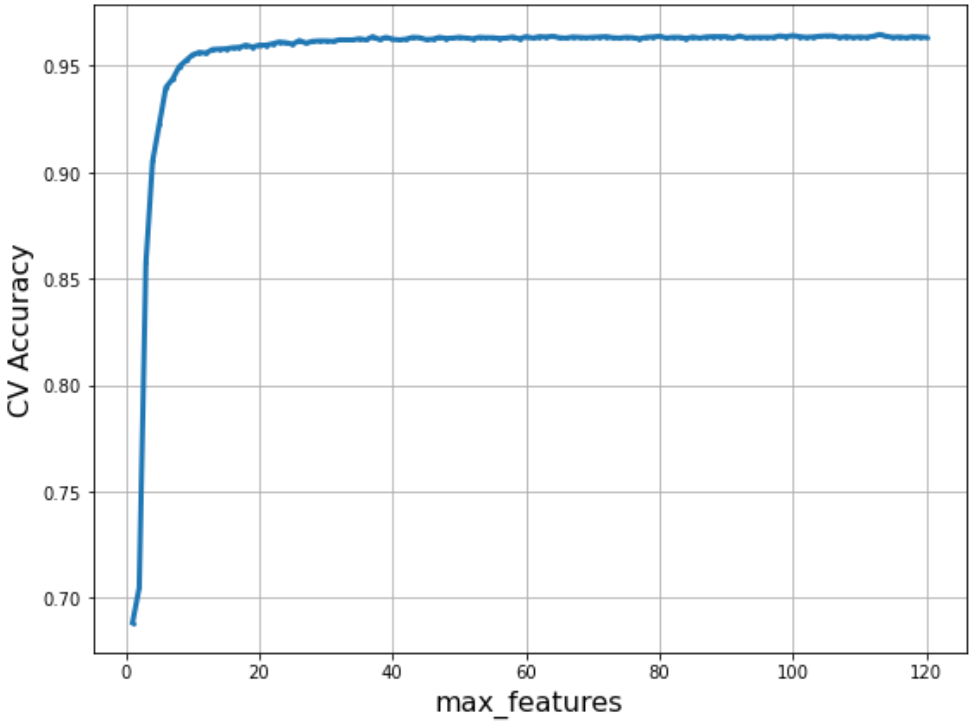
Instructions on how to run the code:

1. Download folder linked above.
2. The .ipynb file reflects the code, which can be run on a Python Jupyter notebook. It shows all modeling and testing.
3. The .csv file shows the dataset, which is read in the code.
4. The PDF file is a PDF version of the code for easier interpretation.

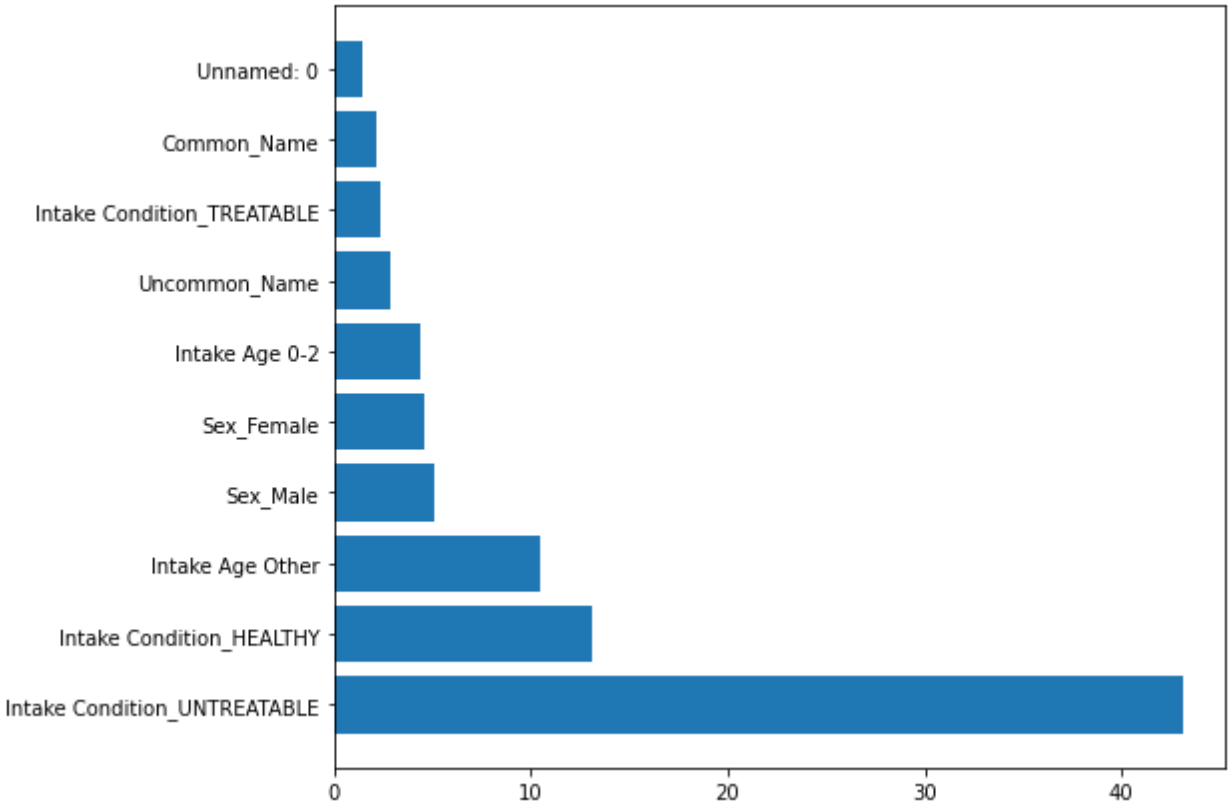
Table C.8: Random Forest Test-set Metrics

Model	Test Accuracy	TPR	FPR	TNR	FNR
Baseline	68.83%	100%	100%	0%	0%
Random Forest with No CV	92.14%	98.52%	21.96%	78.04%	1.48%
Random Forest with CV	95.85%	98.52	10.04%	89.96%	1.48%

Graph C.9: Random Forest with CV Model Max Features Cross-Validated Accuracy



Graph C.10: Random Forest with CV Model Feature Importance



Gradient Boosting Model

Code for the Gradient Boosting modeling, data used, and test metrics can be found here.

Instructions on how to run the code:

1. Download folder linked above.
2. The .ipynb file reflects the code, which can be run on a Python Jupyter notebook. It shows all modeling and testing.
3. The .csv file shows the dataset, which is read in the code.
4. The PDF file is a PDF version of the code for easier interpretation.

Table C.11: Gradient Boosting Test-set Metrics

Model	Test Accuracy	TPR	FPR	TNR	FNR
Baseline	68.83%	100%	100%	0%	0%
Gradient Boosting with No CV	95.36%	97.78%	8.79%	91.21%	5.10%
Gradient Boosting with n_estimators CV	96.36%	98.69%	8.79%	91.21%	1.31%