Classification and Logistic Regression

August 3, 2024

Table of contents

Motivation 1
Objectives 2
The Loans Dataset 2
A New Model Class 2
Building, Reducing, and Interpreting a Logistic Regressor 3
Assessing Classifier performance 16
Summary 16


```
library(tidyverse)
library(tidymodels)
library(patchwork)
library(kableExtra)
library(modeldata)

tidymodels_prefer()

options(kable_styling_bootstrap_options = c("hover", "striped"))

theme_set(theme_bw(base_size = 14))

loans <- read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Loan-Prediction-Datas
```

Motivation

All semester long we've been working with *regression* models. These are models designed to output a *numeric* response. It is also possible to construct models which will predict a *categorical* response. That is the topic of MAT434: Statistical Learning and Classification, but we'll introduce the idea here.

Objectives

In this notebook, we'll accomplish the following:

- Distinguish between *regression* tasks and *classification* tasks.
- Use the `tidymodels` framework to fit and assess a classifier.

The Loans Dataset

We'll be working with a dataset on loans and attempting to build a model which will predict whether a loan is approved or denied. The first few rows of that dataset can be seen below.

```
loans %>%
  head() %>%
  kable() %>%
  kable_styling()
```

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
LP001002	Male	No	0	Graduate	No	5849	
LP001003	Male	Yes	1	Graduate	No	4583	
LP001005	Male	Yes	0	Graduate	Yes	3000	
LP001006	Male	Yes	0	Not Graduate	No	2583	
LP001008	Male	No	0	Graduate	No	6000	
LP001011	Male	Yes	2	Graduate	Yes	5417	

The `Loan_Status` variable is the one we'd like to predict.

A New Model Class

There are two major types of classification model. One simply outputs a class membership prediction, while the other outputs the *propensity* or *likelihood* of a model belonging to a paraticular class of interest. We'll focus on the latter here because outputting a *propensity* is similar to a regression task – we are building a model whose output can be interpreted as a probability. Unfortunately, our linear regression models can't be used for this purpose though because they are *polynomial* models. All polynomials which are not constant are unbounded – that is, we can't construct a linear regression model which will only output responses between 0 and 1. Instead, we'll use a new class of model called *logistic regressors*. A logistic regression model is useful for differentiating between two classes (*binary classification*), and takes the following form:

$$\mathbb{P}[y=1] = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_k x_k}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_k x_k}}$$

The output of a logistic regressor can be interpreted as the probability that the corresponding observation belongs to the class whose label is encoded by 1.

Building, Reducing, and Interpreting a Logistic Regressor

As usual, we'll begin by splitting our data into a training and test set and then further splitting our training data into cross-validation folds. We'll stratify by `Loan_Status` to ensure that there is proportional representation of approved and denied loans in each of our sets.

```
set.seed(123)
loans_split <- initial_split(loans, prop = 0.9, strata = Loan_Status)

loans_train <- training(loans_split)
loans_test <- testing(loans_split)

loans_folds <- vfold_cv(loans_train, v = 5, strata = Loan_Status)
```

Now we'll build a *logistic regression* model specification and a corresponding recipe. We'll package the model and recipe together into a workflow, and then fit the workflow.

```
log_clf <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_cv <- log_wf %>%
  fit_resamples(loans_folds)
```

```
> A | warning: prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
There were issues with some computations   A: x1
There were issues with some computations   A: x1
```

```
log_clf_cv %>%
  collect_metrics() %>%
  kable() %>%
  kable_styling()
```

.metric	.estimator	mean	n	std_err	.config
accuracy	binary	0.8050975	5	0.0157151	Preprocessor1_Model1
brier_class	binary	0.1579879	5	0.0136477	Preprocessor1_Model1
roc_auc	binary	0.7432406	5	0.0365422	Preprocessor1_Model1

```
log_clf_cv %>%
  collect_metrics(summarize = FALSE) %>%
  kable() %>%
  kable_styling()
```

id	.metric	.estimator	.estimate	.config
Fold1	accuracy	binary	0.7551020	Preprocessor1_Model1
Fold1	roc_auc	binary	0.6432354	Preprocessor1_Model1
Fold1	brier_class	binary	0.2021081	Preprocessor1_Model1
Fold2	accuracy	binary	0.8383838	Preprocessor1_Model1
Fold2	roc_auc	binary	0.8130930	Preprocessor1_Model1
Fold2	brier_class	binary	0.1309485	Preprocessor1_Model1
Fold3	accuracy	binary	0.8020833	Preprocessor1_Model1
Fold3	roc_auc	binary	0.7320099	Preprocessor1_Model1
Fold3	brier_class	binary	0.1583560	Preprocessor1_Model1
Fold4	accuracy	binary	0.8387097	Preprocessor1_Model1
Fold4	roc_auc	binary	0.8375661	Preprocessor1_Model1
Fold4	brier_class	binary	0.1281081	Preprocessor1_Model1
Fold5	accuracy	binary	0.7912088	Preprocessor1_Model1
Fold5	roc_auc	binary	0.6902985	Preprocessor1_Model1
Fold5	brier_class	binary	0.1704186	Preprocessor1_Model1

The accuracy of the model is hovering at around 80%. Let's fit the model to our training data and interpret the model coefficients.

```
log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  glance() %>%
  kable() %>%
  kable_styling()
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
589.2224	476	-214.6937	467.3875	546.5703	429.3875	458	477

```
log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-2.4989186	0.9610571	-2.6001770	0.0093176
ApplicantIncome	0.0000232	0.0000362	0.6389763	0.5228383
CoapplicantIncome	-0.0000431	0.0000438	-0.9849162	0.3246653
LoanAmount	-0.0020185	0.0020136	-1.0024311	0.3161354
Loan_Amount_Term	-0.0015120	0.0020646	-0.7323285	0.4639681
Credit_History	3.8222216	0.4743907	8.0571180	0.0000000
Gender_Male	0.3167624	0.3372321	0.9393010	0.3475762
Gender_other	-0.0317584	0.8427894	-0.0376825	0.9699409
Married_Yes	0.5112299	0.2965235	1.7240789	0.0846936
Married_other	13.1231566	882.7439373	0.0148663	0.9881388
Dependents_X1	-0.4114211	0.3562345	-1.1549164	0.2481246
Dependents_X2	0.2016477	0.3794333	0.5314443	0.5951109
Dependents_X3.	0.2989076	0.5055901	0.5912055	0.5543828
Dependents_other	-0.1496525	0.9109907	-0.1642745	0.8695151
Education_Not.Graduate	-0.5076586	0.3055898	-1.6612419	0.0966649
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1989879	0.3484903	-0.5709996	0.5679999
Self_Employed_unknown	1.4004095	0.7923403	1.7674344	0.0771555
Property_Area_Semiurban	0.8870297	0.3075344	2.8843262	0.0039225
Property_Area_Urban	0.0666169	0.3044695	0.2187966	0.8268085

Property_Area_other	NA	NA	NA	NA

Like linear regression, we get a p-value associated with each model term. We can use this information to remove predictors from the model if there is not statistically significant evidence to suggest that they are useful. In the output above, we can see that `ApplicantIncome` has a high p-value – we'll take care of that one first since the p-values which are higher are each attached to a single level of a categorical variable.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-2.4529843	0.9599955	-2.5552039	0.0106126
CoapplicantIncome	-0.0000514	0.0000423	-1.2147592	0.2244579
LoanAmount	-0.0011197	0.0014552	-0.7694332	0.4416362
Loan_Amount_Term	-0.0016404	0.0020650	-0.7943544	0.4269892
Credit_History	3.8203641	0.4733598	8.0707407	0.0000000
Gender_Male	0.3110792	0.3372429	0.9224187	0.3563102
Gender_other	0.0228067	0.8360966	0.0272776	0.9782383
Married_Yes	0.4990121	0.2960870	1.6853565	0.0919198
Married_other	13.1264249	882.7439371	0.0148700	0.9881359
Dependents_X1	-0.4196812	0.3550546	-1.1820186	0.2371983

Dependents_X2	0.1906272	0.3783393	0.5038527	0.6143649
Dependents_X3.	0.3273592	0.5051927	0.6479887	0.5169922
Dependents_other	-0.1728850	0.9104306	-0.1898937	0.8493925
Education_Not.Graduate	-0.5105283	0.3053041	-1.6721959	0.0944857
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1660175	0.3439122	-0.4827323	0.6292858
Self_Employed_unknown	1.4212648	0.7929617	1.7923498	0.0730769
Property_Area_Semiurban	0.8983514	0.3070095	2.9261359	0.0034320
Property_Area_Urban	0.0907199	0.3019130	0.3004837	0.7638082
Property_Area_other	NA	NA	NA	NA

We'll now remove `LoanAmount` due to its high p-value.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-2.6973736	0.9231379	-2.9219617	0.0034783
CoapplicantIncome	-0.0000495	0.0000422	-1.1742760	0.2402845
Loan_Amount_Term	-0.0012141	0.0019815	-0.6127143	0.5400653
Credit_History	3.7968026	0.4679064	8.1144487	0.0000000
Gender_Male	0.2625179	0.3328265	0.7887531	0.4302563

Gender_other	-0.0376062	0.8277210	-0.0454334	0.9637618
Married_Yes	0.4694940	0.2865163	1.6386294	0.1012905
Married_other	12.8605374	624.1471958	0.0206050	0.9835608
Dependents_X1	-0.3183561	0.3465245	-0.9187117	0.3582464
Dependents_X2	0.1710928	0.3670923	0.4660757	0.6411613
Dependents_X3.	0.1642606	0.4743118	0.3463135	0.7291071
Dependents_other	0.0480555	0.8662543	0.0554751	0.9557600
Education_Not.Graduate	-0.5009039	0.2886666	-1.7352331	0.0826995
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1290061	0.3378401	-0.3818557	0.7025684
Self_Employed_unknown	1.4571460	0.7963133	1.8298653	0.0672701
Property_Area_Semiurban	0.8638045	0.3006735	2.8728981	0.0040673
Property_Area_Urban	0.0228666	0.2911781	0.0785313	0.9374054
Property_Area_other	NA	NA	NA	NA

And now we'll remove the `Loan_Amount_Term` predictor.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.1896069	0.5689535	-5.6060939	0.0000000
CoapplicantIncome	-0.0000466	0.0000418	-1.1156774	0.2645603
Credit_History	3.8889114	0.4639547	8.3820936	0.0000000
Gender_Male	0.2287428	0.3304710	0.6921721	0.4888292
Gender_other	-0.0590459	0.8286520	-0.0712554	0.9431945
Married_Yes	0.4780451	0.2857684	1.6728407	0.0943587
Married_other	12.7888198	623.2316618	0.0205202	0.9836284
Dependents_X1	-0.3385201	0.3425413	-0.9882605	0.3230251
Dependents_X2	0.1390872	0.3668929	0.3790949	0.7046174
Dependents_X3.	0.1460591	0.4715114	0.3097679	0.7567375
Dependents_other	0.1089389	0.8480836	0.1284531	0.8977904
Education_Not.Graduate	-0.4291522	0.2829370	-1.5167767	0.1293231
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1016076	0.3360384	-0.3023690	0.7623708
Self_Employed_unknown	1.4951455	0.7970839	1.8757693	0.0606870
Property_Area_Semiurban	0.8580720	0.3001088	2.8592030	0.0042471
Property_Area_Urban	0.0689477	0.2863381	0.2407913	0.8097169
Property_Area_other	NA	NA	NA	NA

Now `Gender` gets removed.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
```

```
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.0519837	0.5275751	-5.7849273	0.0000000
CoapplicantIncome	-0.0000421	0.0000412	-1.0240013	0.3058347
Credit_History	3.8901229	0.4633957	8.3948184	0.0000000
Married_Yes	0.5410979	0.2690896	2.0108466	0.0443417
Married_other	12.7688124	624.1944502	0.0204565	0.9836792
Dependents_X1	-0.3567241	0.3412027	-1.0454903	0.2957964
Dependents_X2	0.1495488	0.3664164	0.4081391	0.6831716
Dependents_X3.	0.1468830	0.4697043	0.3127138	0.7544981
Dependents_other	0.1174474	0.8483378	0.1384441	0.8898894
Education_Not.Graduate	-0.4157659	0.2815194	-1.4768640	0.1397121
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1113688	0.3341613	-0.3332785	0.7389241
Self_Employed_unknown	1.5107580	0.7982406	1.8926099	0.0584098
Property_Area_Semiurban	0.8416688	0.2987829	2.8169912	0.0048476
Property_Area_Urban	0.0707943	0.2860912	0.2474535	0.8045573
Property_Area_other	NA	NA	NA	NA

And now the `CoapplicantIncome`.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_rm(CoapplicantIncome) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)
```

```
log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.1117019	0.5247893	-5.9294312	0.0000000
Credit_History	3.8831039	0.4627203	8.3919035	0.0000000
Married_Yes	0.5152269	0.2674210	1.9266507	0.0540232
Married_other	12.8044060	624.1944485	0.0205135	0.9836338
Dependents_X1	-0.3314454	0.3397146	-0.9756585	0.3292337
Dependents_X2	0.1486167	0.3660850	0.4059623	0.6847703
Dependents_X3.	0.1871269	0.4668371	0.4008397	0.6885381
Dependents_other	0.1449574	0.8470528	0.1711315	0.8641204
Education_Not.Graduate	-0.3917205	0.2801854	-1.3980758	0.1620903
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1132400	0.3341939	-0.3388451	0.7347264
Self_Employed_unknown	1.5284863	0.8012526	1.9076210	0.0564402
Property_Area_Semiurban	0.8453024	0.2986458	2.8304513	0.0046482
Property_Area_Urban	0.0750451	0.2857465	0.2626282	0.7928372
Property_Area_other	NA	NA	NA	NA

Now the `Dependents` variable should be removed since all of its levels have p-values higher than the other model terms.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_rm(CoapplicantIncome) %>%
  step_rm(Dependents) %>%
  step_other(all_nominal_predictors()) %>%
```

```
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.0711821	0.5151412	-5.9618256	0.0000000
Credit_History	3.8458190	0.4553814	8.4452703	0.0000000
Married_Yes	0.5209900	0.2385411	2.1840676	0.0289573
Married_other	12.9738410	624.1938874	0.0207850	0.9834172
Education_Not.Graduate	-0.3834507	0.2790899	-1.3739324	0.1694627
Education_other	NA	NA	NA	NA
Self_Employed_Yes	-0.1339977	0.3310414	-0.4047763	0.6856419
Self_Employed_unknown	1.5187712	0.8067847	1.8824988	0.0597683
Property_Area_Semiurban	0.8175899	0.2969759	2.7530507	0.0059043
Property_Area_Urban	0.0417614	0.2805583	0.1488510	0.8816712
Property_Area_other	NA	NA	NA	NA

And now `Education`.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_rm(CoapplicantIncome) %>%
  step_rm(Dependents) %>%
```

```
  step_rm(Education) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.2117754	0.5067024	-6.3385829	0.0000000
Credit_History	3.8733275	0.4542484	8.5268927	0.0000000
Married_Yes	0.5249213	0.2383121	2.2026636	0.0276185
Married_other	13.0574215	624.1938841	0.0209189	0.9833104
Self_Employed_Yes	-0.1239209	0.3307729	-0.3746404	0.7079279
Self_Employed_unknown	1.5154531	0.8119814	1.8663643	0.0619904
Property_Area_Semiurban	0.8470942	0.2959740	2.8620563	0.0042090
Property_Area_Urban	0.0911532	0.2772758	0.3287457	0.7423479
Property_Area_other	NA	NA	NA	NA

Next we'll remove the `Self_Employed` variable.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_rm(CoapplicantIncome) %>%
  step_rm(Dependents) %>%
  step_rm(Education) %>%
```

```
  step_rm(Self_Employed) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.1494152	0.4971941	-6.3343778	0.0000000
Credit_History	3.8671654	0.4485669	8.6211572	0.0000000
Married_Yes	0.4981902	0.2364029	2.1073776	0.0350849
Married_other	13.0088663	624.1938810	0.0208411	0.9833724
Property_Area_Semiurban	0.8394513	0.2949382	2.8461940	0.0044245
Property_Area_Urban	0.0844338	0.2745243	0.3075638	0.7584142
Property_Area_other	NA	NA	NA	NA

Now, all of the predictors are attached to p-values below the 5% threshold for statistical significance. Note that the high p-value attached to `Property_Area_Urban` indicates no evidence for that particular level of the `Property_Area` having a different effect on loan approval or denial than the base level of `Rural`. The property area being *semiurban* though is associated with a higher likelihood of loan approval, which we can tell because its coefficient is *positive*. We were unable to estimate the impact of having a *missing* property area because there were so few observations in this category. We can similarly interpret the other coefficients.

- Better (higher) `Credit_History` is associated with a higher likelihood of loan approval.
- Applicants who are `Married` have a higher likelihood of loan approval than those who were not married or those whose marital status was unknown.
- The coefficient fit for loan applications in an *Urban* `Property_Area` indicates a higher likelihood of loan approval than applicants in a *Rural* `Property_Area` – the high p-value, however, indicates no statistical evidence to support this.

- The coefficient fit for loan applications in an *Semiurban* `Property_Area` indicates a higher likelihood of loan approval than applicants in a *Rural* `Property_Area`.
- No coefficient could be fit for loan applications whose `Property_Area` was unlisted because there were 0 records missing this information in the training set.

Since there are no observations with missing `Property_Area`, we'll refit the model by removing that predictor.

```
log_rec <- recipe(Loan_Status ~ ., data = loans_train) %>%
  step_rm(Loan_ID) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_rm(ApplicantIncome) %>%
  step_rm(LoanAmount) %>%
  step_rm(Loan_Amount_Term) %>%
  step_rm(Gender) %>%
  step_rm(CoapplicantIncome) %>%
  step_rm(Dependents) %>%
  step_rm(Education) %>%
  step_rm(Self_Employed) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_rm(Property_Area_other)

log_wf <- workflow() %>%
  add_model(log_clf) %>%
  add_recipe(log_rec)

log_clf_fit <- log_wf %>%
  fit(loans_train)

log_clf_fit %>%
  extract_fit_engine() %>%
  tidy() %>%
  kable() %>%
  kable_styling()
```

term	estimate	std.error	statistic	p.value
Intercept	-3.1494152	0.4971941	-6.3343778	0.0000000
Credit_History	3.8671654	0.4485669	8.6211572	0.0000000
Married_Yes	0.4981902	0.2364029	2.1073776	0.0350849
Married_other	13.0088663	624.1938810	0.0208411	0.9833724

Property_Area_Semiurban	0.8394513	0.2949382	2.8461940	0.0044245
Property_Area_Urban	0.0844338	0.2745243	0.3075638	0.7584142

Assessing Classifier performance

The metrics we've been depending on, R-Squared and the Root Mean Squared Error, are meaningless for classification. There are lots of classification performance metrics, but we'll focus on the simplest one here – *accuracy*. This metric is exactly what it sounds like – out of all the predictions we made, what proportion were correct? Let's assess our model's accuracy on the test data below.

```
log_clf_fit %>%
  augment(loans_test) %>%
  select(Loan_Status, .pred_class) %>%
  filter(!is.na(.pred_class)) %>%
  summarize(accuracy = sum(Loan_Status == .pred_class)/ n()) %>%
  kable() %>%
  kable_styling()
```

accuracy
0.8596491

Our model achieved 85.7% accuracy on the unseen test data! We won't know whether this is good or bad until we try building other models to beat this one. For now, since 68.3% of all loans were approved, our model is doing better than just naively guessing that every loan will be approved.

Summary

This notebook introduced the notion of classification – building models to predict a categorical outcome (such as whether or not a loan will be approved). We saw that the same `{tidymodels}` framework we've utilized to approach *regression* tasks was also useful for *classification* tasks. If you'd like to learn more about classification, I'd love to see you in *MAT434: Statistical Learning and Classification* next semester!