Machine Learning

German credit score & NYC Taxi Fare Prediction

Regression - NYC Taxi fare

Executive Summary

10.44

CV MAPE

9.77

Test MAPE

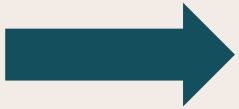
Random Forest

Best Model?

Used Variables

Initial Variables

- ✗ dropoff_latitude
- ✗ dropoff_longitude
- ✗ fare_amount
- ✗ features from feat01 to feat10
- ✗ key
- ✗ passenger_count
- ✗ pickup_datetime
- ✗ pickup_latitude
- ✗ pickup_longitude

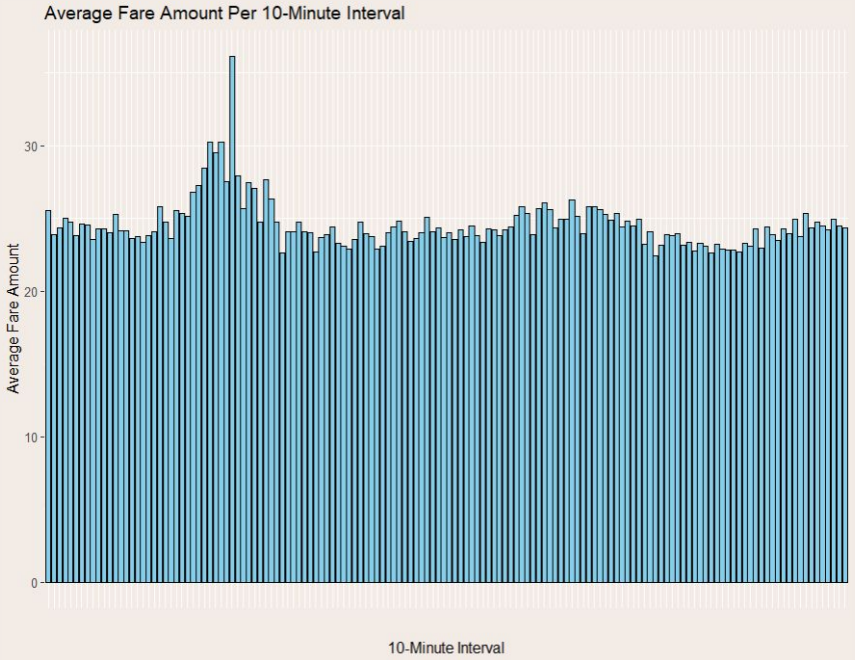


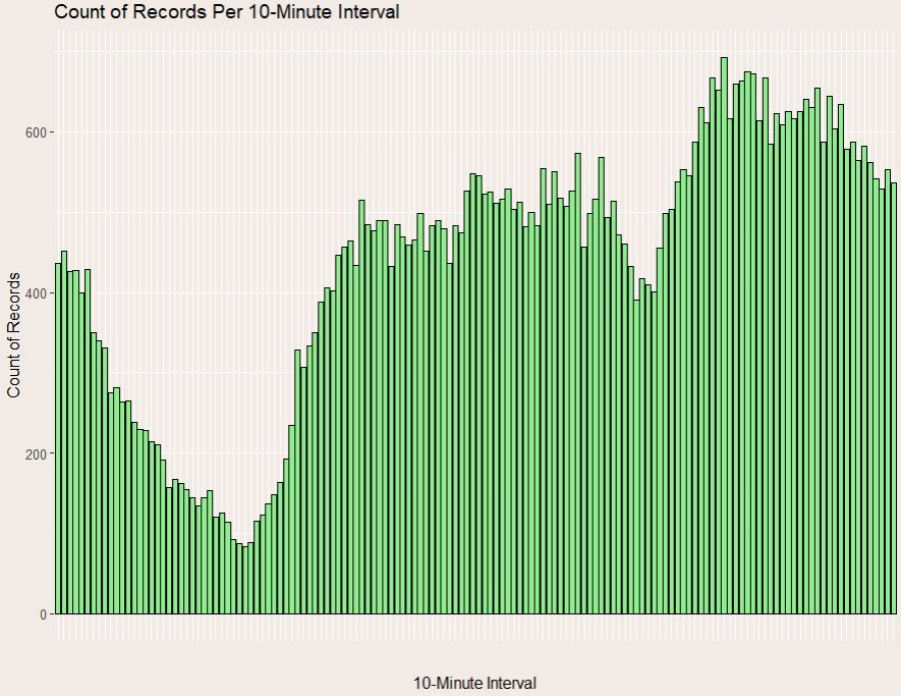
Final Variables

- ✓ dropoff_latitude
- ✓ dropoff_longitude
- ✓ fare_amount
- ✓ features from feat01 to feat10
- ✓ passenger_count
- ✓ pickup_latitude
- ✓ pickup_longitude
- ✓ zero_indicator
- ✓ straight_dist
- ✓ count
- ✓ hour4
- ✓ hour5
- ✓ hour19
- ✓ hour20

Feature Exploratory Data Analysis

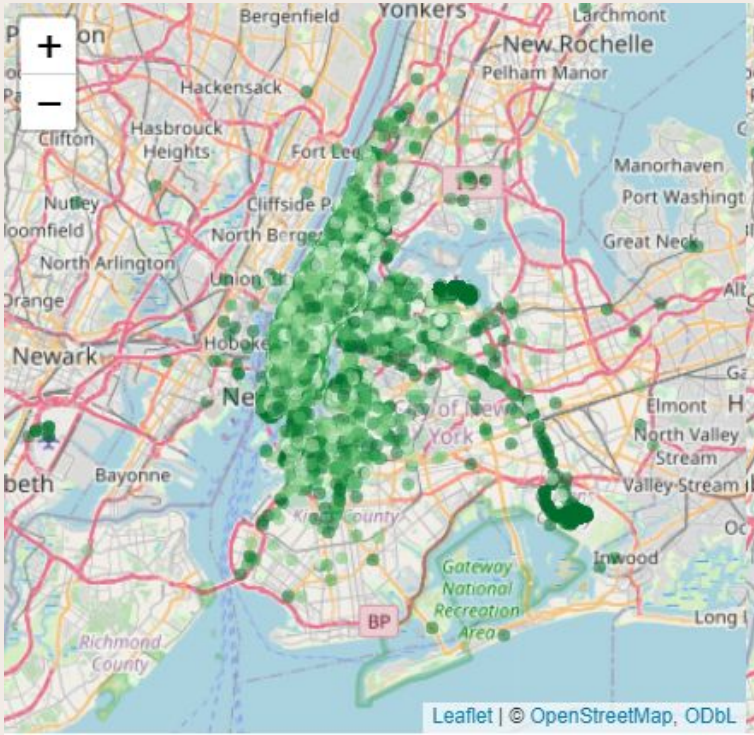
Time dependancy



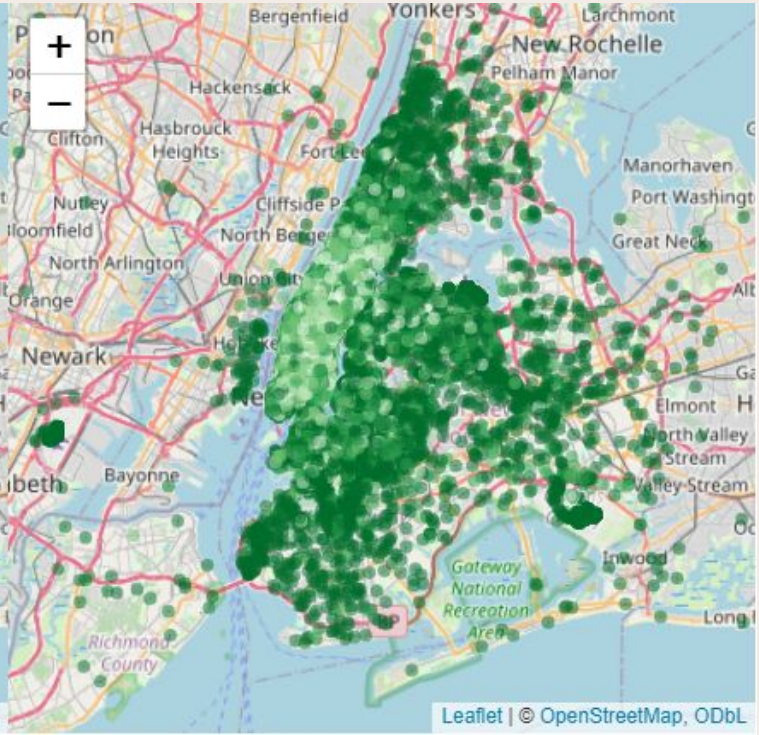


Feature Exploratory Data Analysis

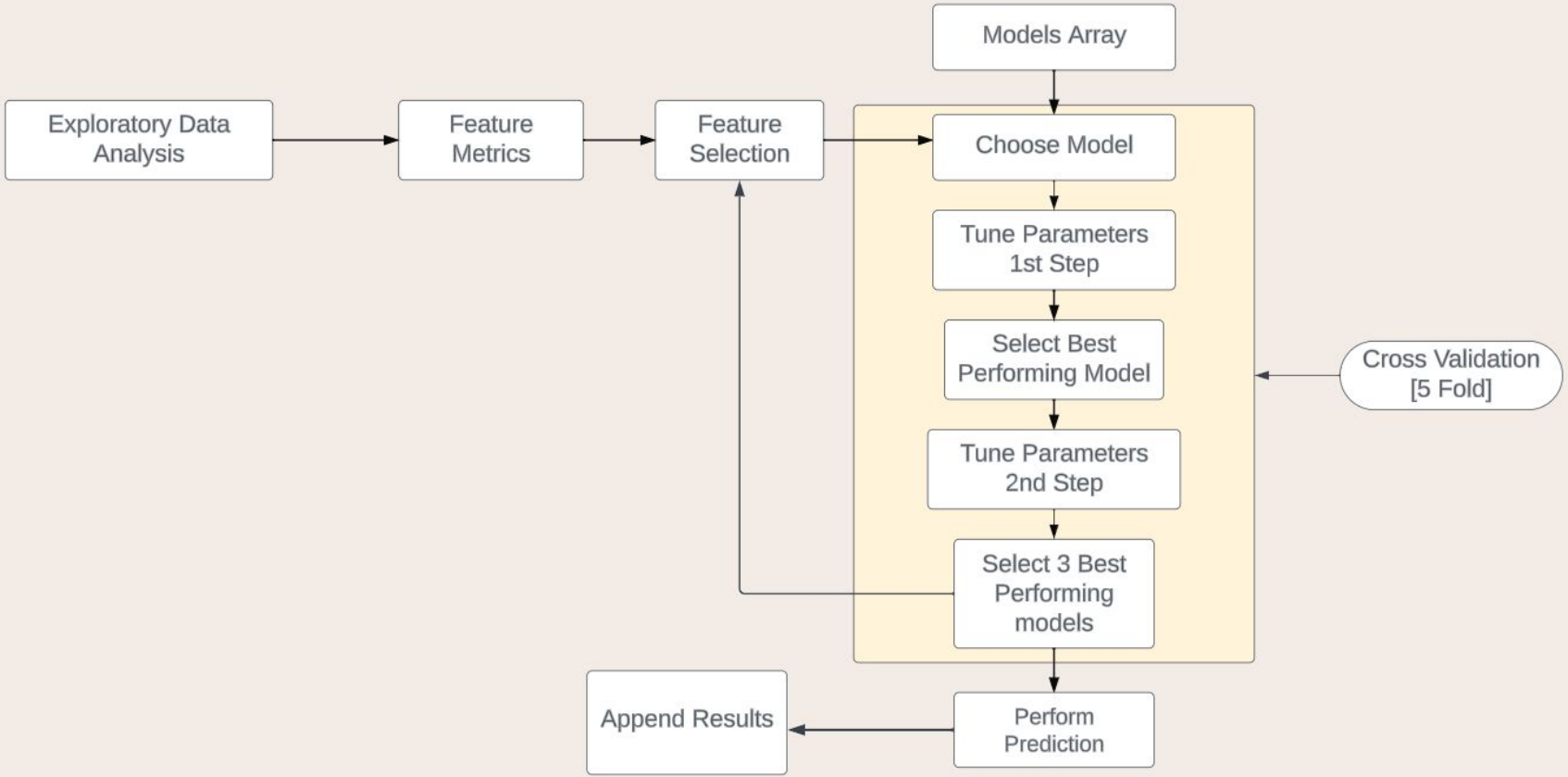
Pickup localization



Dropoff localization



Model of dataflow

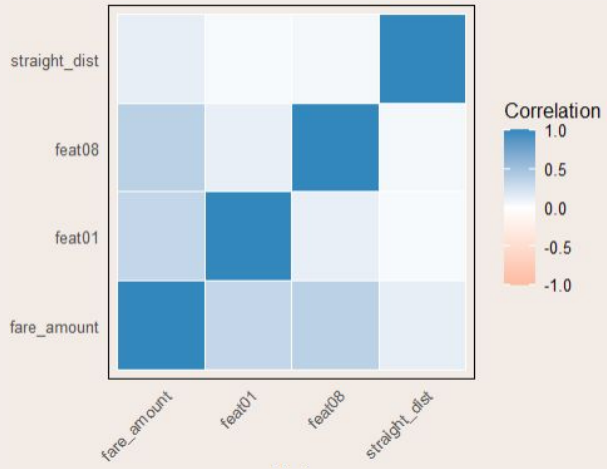


Feature Selection

1

Mutual Information

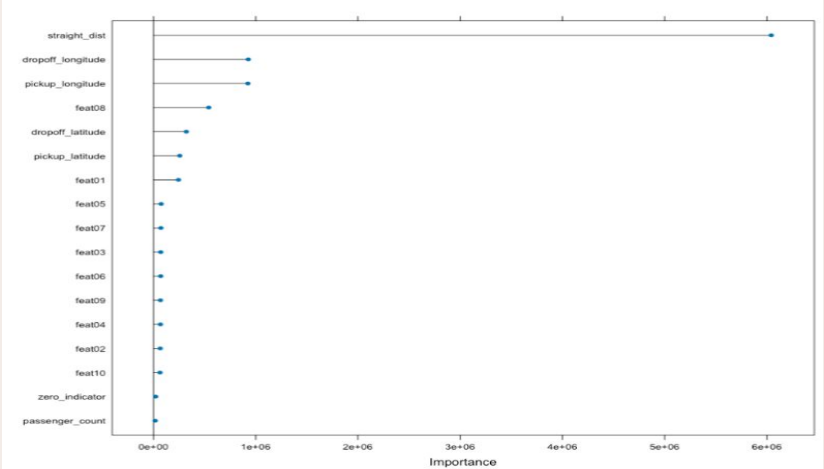
2 **Correlation**



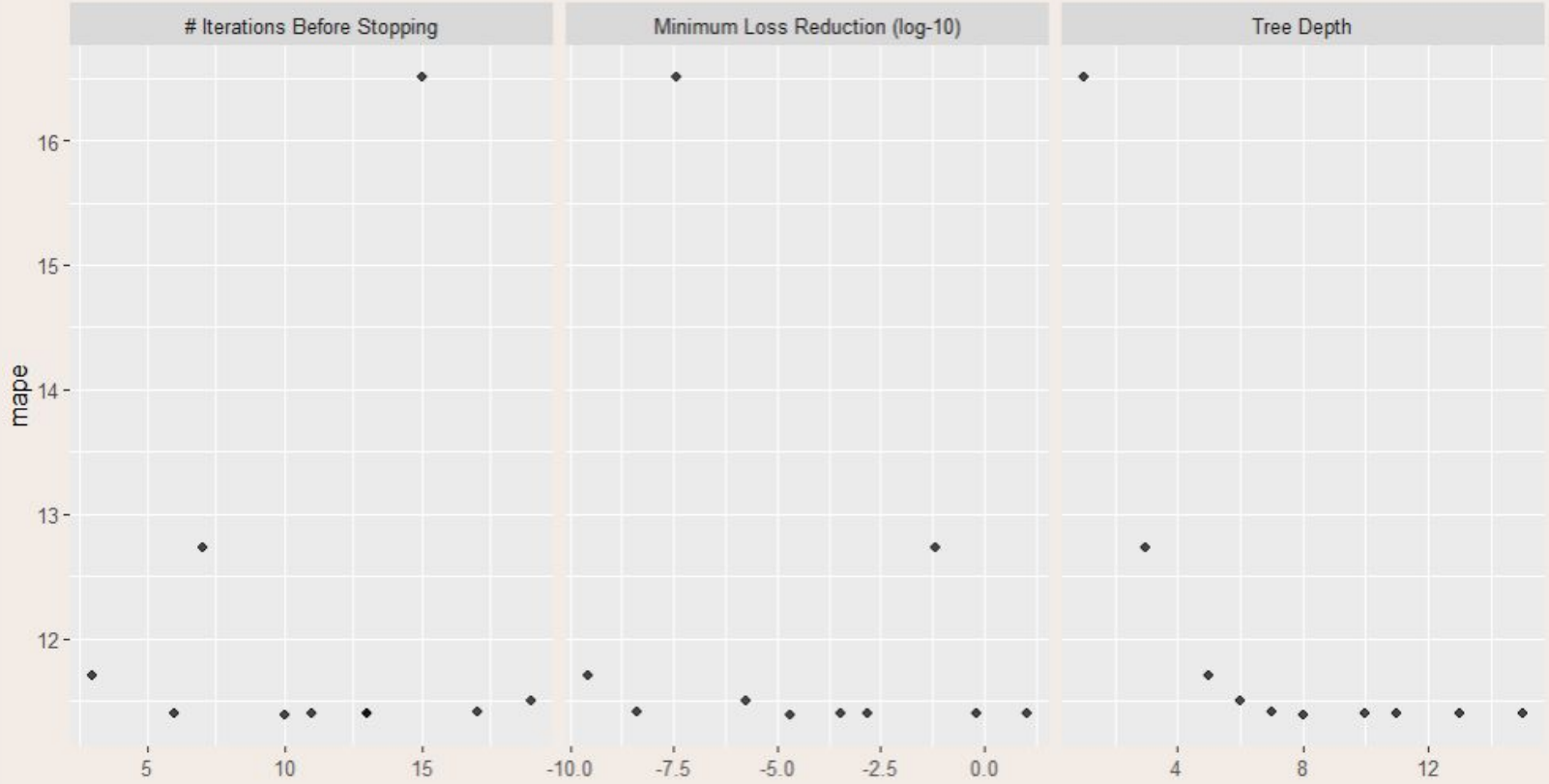
3

All Values

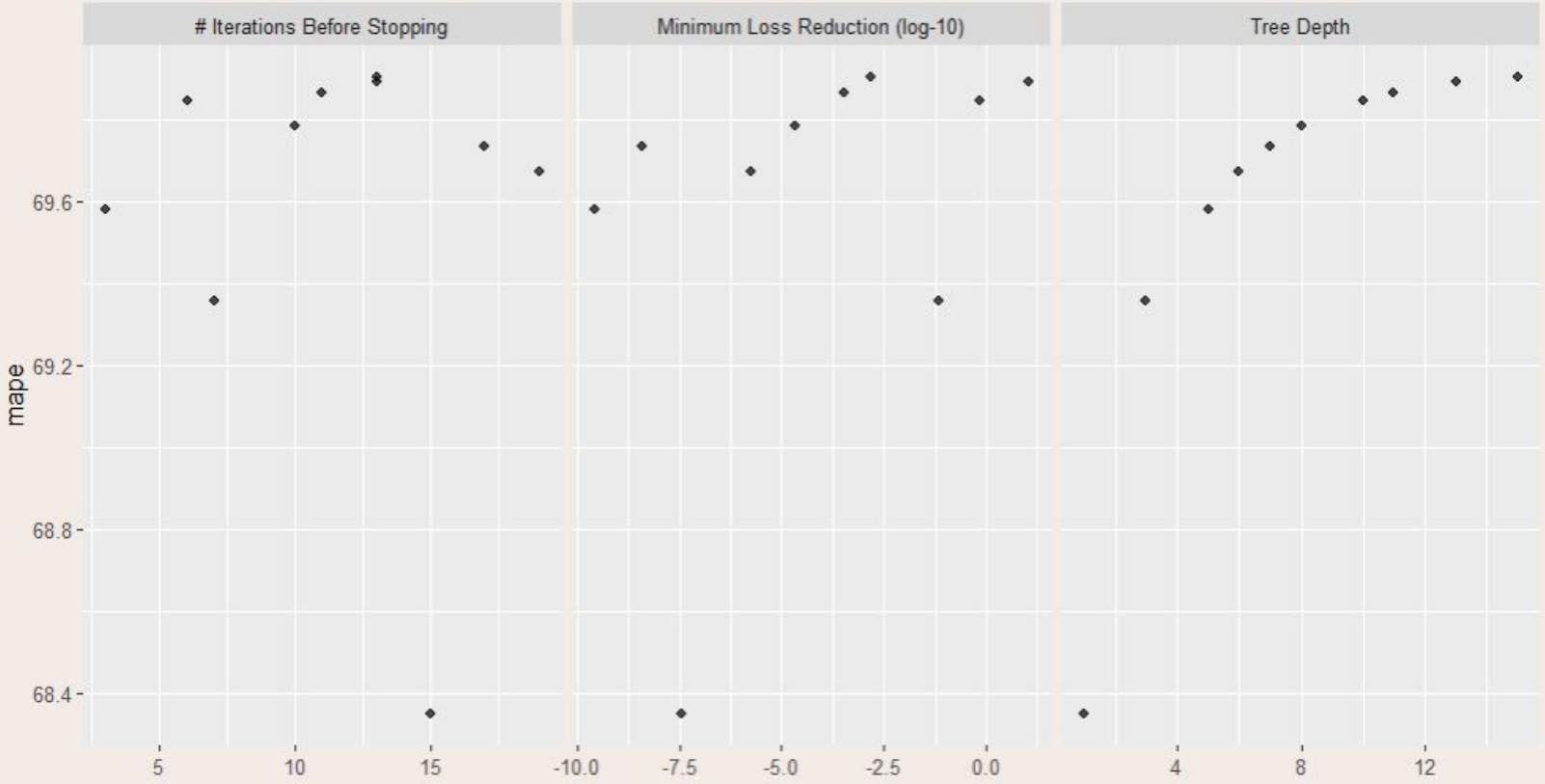
4 **Importance Score**



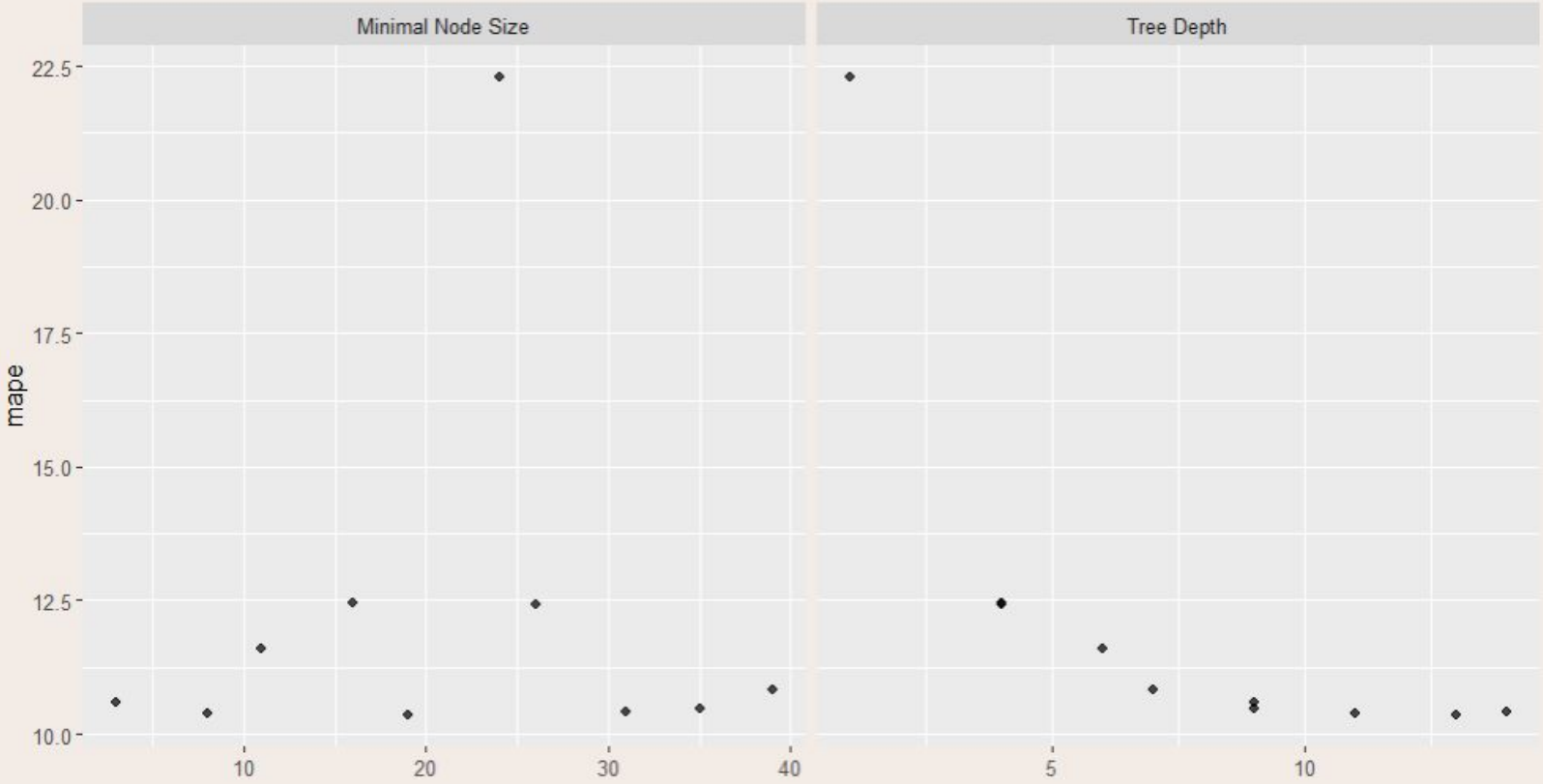
LGBM Tuning


Iterations Before Stopping
Minimum Loss Reduction (log-10)
Tree Depth
mape
16
15
14
13
12
5
10
15
-10.0
-7.5
-5.0
-2.5
0.0
4
8
12

XGBoost Tuning



Decision Tree Tuning



Results

Iteration	Mape_Train	Mape_test	models_spec	model_name
3	10,40	9,77	trees = 853, min_n = 9, mtry = 6	random_forest
1	10,37	9,77	trees = 853, min_n = 5, mtry = 6	random_forest
2	10,37	9,78	trees = 853, min_n = 15, mtry = 6	random_forest
2	10,37	10,58	min_n = 8, tree_depth = 11, cost_complexity = 0	decision_tree
1	10,37	10,84	min_n = 19, tree_depth = 13, cost_complexity = 0	decision_tree
3	10,40	10,94	min_n = 31, tree_depth = 14, cost_complexity = 0	decision_tree
1	12,33	11,48	learn_rate = 0.02, tree_depth = 8, loss_reduction = 0.97, stop_iter = 18	lgbm
2	12,34	11,52	learn_rate = 0.02, tree_depth = 9, loss_reduction = 0, stop_iter = 19	lgbm
3	12,34	11,56	learn_rate = 0.02, tree_depth = 11, loss_reduction = 0, stop_iter = 7	lgbm
1	68,35	70,94	learn_rate = 0.02, tree_depth = 2, loss_reduction = 0.01, stop_iter = 11	xgboost
2	69,36	71,11	learn_rate = 0.02, tree_depth = 3, loss_reduction = 0.02, stop_iter = 9	xgboost
3	69,58	71,27	learn_rate = 0.02, tree_depth = 5, loss_reduction = 0, stop_iter = 5	xgboost

Classification problem

Executive Summary

92.1%

CV GINI

77%

Test GINI

XGB

Best Model?

Data cleaning

- Missing values - no missing values
- Personal status column (female div/dep/mar, male div/sep, male mar/wid, male single) - divided into 2 columns: sex, marital_status
- Ordered categorical variables - as on the picture
- Boolean variables
- One hot-encoding applied to all categorical variables (remove_first_dummy = TRUE)

Finally we finished data cleaning process with 51 variables (including id)

explained variable: bad: 30%, good: 70%

```
## credit_history :  existing paid, delayed previously, critical/other existing credit, all paid, no cr
edits/all paid
## housing :  own, for free, rent
## other_parties :  none, guarantor, co applicant
## other_payment_plans :  none, bank, stores
## property_magnitude :  real estate, life insurance, car, no known property
## purpose :  furniture/equipment, new car, radio/tv, used car, education, repairs, business, retrainin
g, other, domestic appliance
## marital_status :  single, div/dep/mar, mar/wid, div/sep
```

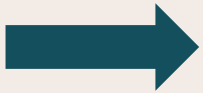
```
checking_status_mapping <- c(
  "no checking" = 1,
  "<0" = 2,
  "0<=X<200" = 3,
  ">=200" = 4
  )

employment_mapping <- c(
  "unemployed" = 1,
  "<1" = 2,
  "1<=X<4" = 3,
  "4<=X<7"= 4,
  ">=7" = 5)

job_mapping <- c(
    "unemp/unskilled non res" = 1,
    "unskilled resident" = 1,
    "skilled" = 2,
    "high qualif/self emp/mgmt" = 3
)

savings_status_mapping <- c(
    "no known savings" = 1,
    "<100" = 2,
    "100<=X<500" = 3,
    "500<=X<1000" = 4,
    ">=1000" = 5
)
```

```
##   class foreign_worker own_telephone
## 1  good            yes          none
## 2  good            yes          none
## 3  good             no          none
## 4  good            yes          none
## 5  good            yes           yes
## 6  good            yes          none
```

Recursive feature elimination

50 variables (without class)

id	feat10	other_payment_plans_none
age	foreign_worker	other_payment_plans_stores
checking_status	installment_commitment	property_magnitude_life.insurance
class	job	property_magnitude_no.known.property
credit_amount	num_dependents	property_magnitude_real.estate
duration	own_telephone	purpose_domestic.appliance
employment	residence_since	purpose_education
existing_credits	savings_status	purpose_furniture.equipment
feat01	sex	purpose_new.car
feat02	credit_history_critical.other.existing.credit	purpose_other
feat03	credit_history_delayed.previously	purpose_radio.tv
feat04	credit_history_existing.paid	purpose_repairs
feat05	credit_history_no.credits.all.paid	purpose_retraining
feat06	housing_own	purpose_used.car
feat07	housing_rent	marital_status_div.sep
feat08	other_parties_guarantor	marital_status_mar.wid
feat09	other_parties_none	marital_status_single



33 final predictors

checking_status	purpose_radio.tv
feat02	credit_history_existing.paid
duration	existing_credits
credit_amount	other_payment_plans_stores
feat10	residence_since
feat01	other_parties_guarantor
age	job
credit_history_critical.other.existing.credit	credit_history_delayed.previously
savings_status	purpose_new.car
property_magnitude_real.estate	own_telephone
employment	housing_rent
other_payment_plans_none	other_parties_none
installment_commitment	purpose_used.car
credit_history_no.credits.all.paid	marital_status_single
property_magnitude_no.known.property	sex
housing_own	property_magnitude_life.insurance
purpose_education	

XGB - one by one

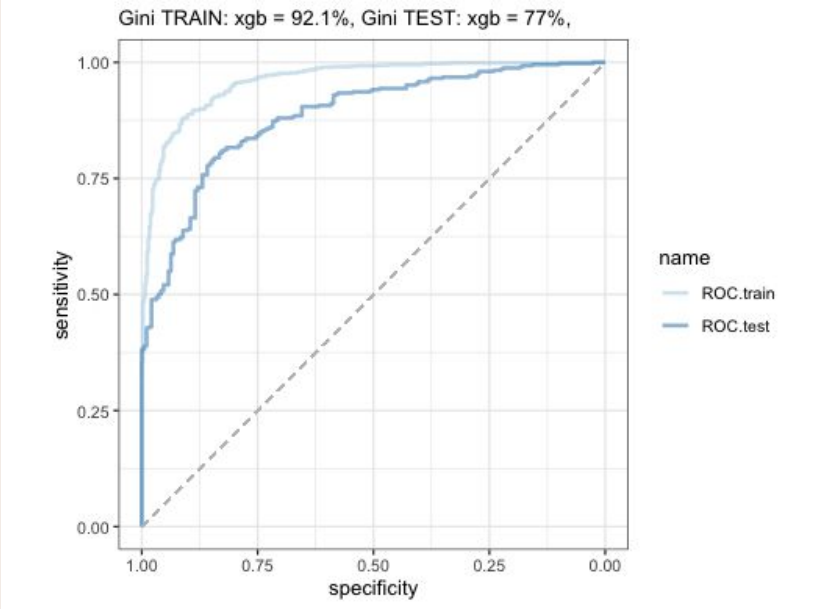
- The order of tuning parameters:
  - nrounds - 40
  - max_depth, min_child_weight (5, 12)
  - colsample_bytree - 0.65
  - subsample - 0.85
  - double nrounds, reduce by half learning rate (80, 0.12)
  - double nrounds, reduce by half learning rate (160, 0.06)

The last model was the best - we choose this one

GINI train: **92.1%**

GINI test: **77%**

- colsample_bytree - rule of thumb(number of predictors - sqrt(35)/35 (0.17)
- min_child_weight - 0.5 - 1% of num of obs (10–20)



	xgb_model.1	xgb_model.2	xgb_model.3	xgb_model.4	xgb_model.5	xgb_model.6
Accuracy	0.81	0.80	0.80	0.81	0.81	0.81
Sensitivity	0.92	0.92	0.90	0.91	0.91	0.91
Specificity	0.58	0.54	0.59	0.59	0.61	0.60
Gini	0.76	0.75	0.75	0.76	0.76	0.77

GBM

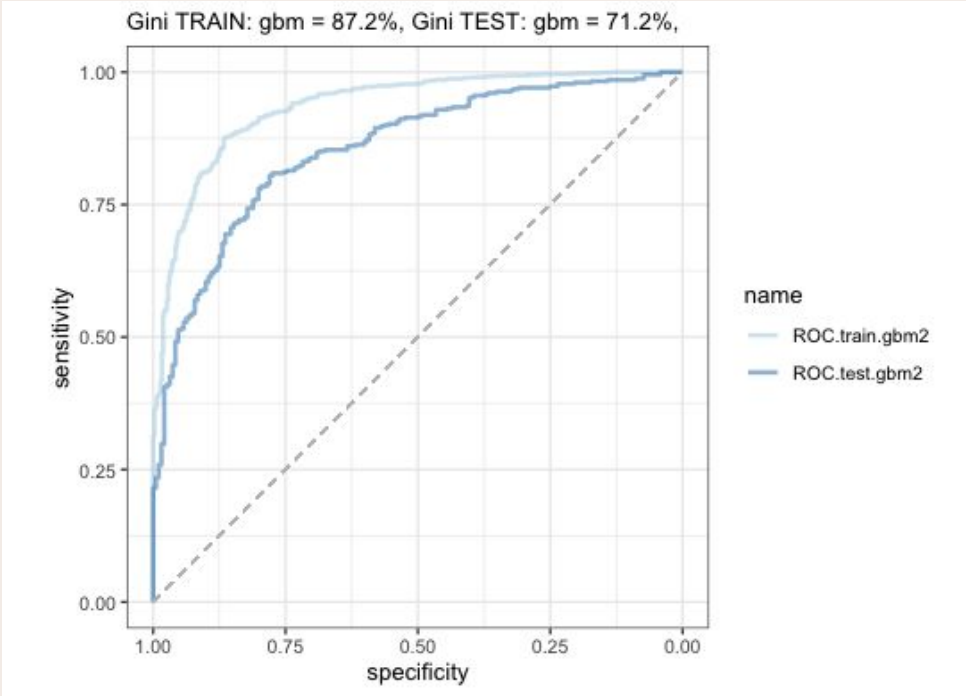
```
expand.grid(interaction.depth = c(1, 2, 4),
            n.trees = c(100, 300, 500),
            shrinkage = c(0.01, 0.1),
            n.minobsinnode = c(150, 250, 400))
```

- Hyperparameter tuning - the best for this model
  - n.trees = 500
  - interaction.depth = 4
  - shrinkage = 0.1
  - n.minobsinnode = 150

GINI train: **84.3%**

GINI test: **74.8%**

	Accuracy	Sensitivity	Specificity	Gini
Train	0.88	0.94	0.72	0.87
Test	0.79	0.88	0.59	0.71



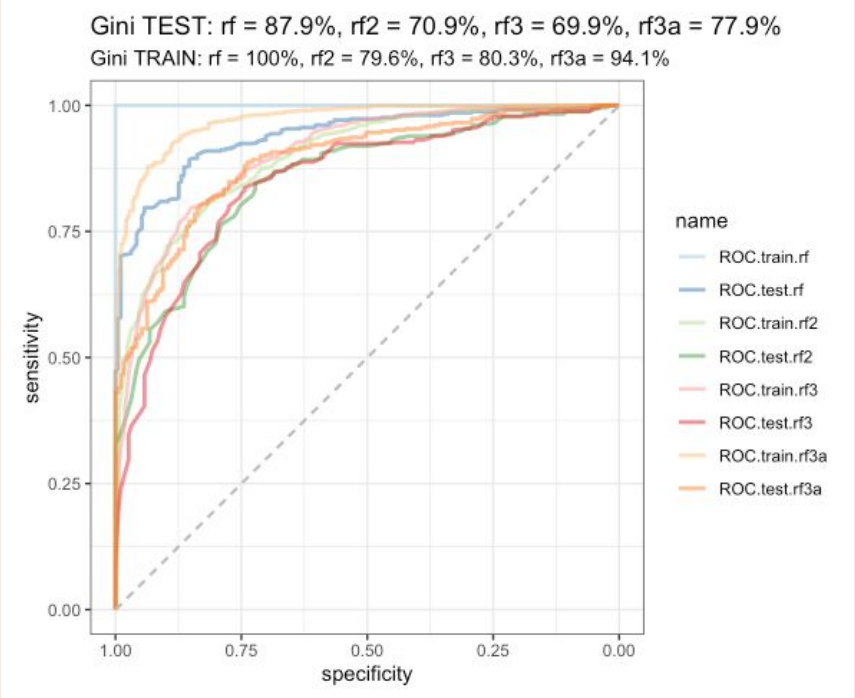
Random Forest

- Models with and without cv
- Hyperparameter tuning - the best for this model:
  - mtry = 6
  - min.node.size = 50

GINI train: **94.1%**

GINI test: **77.9%**

```
parameters_ranger <-
  expand.grid(mtry = 4:15,
              # split rule
              splitrule = "gini",
              # minimum size of the terminal node
              min.node.size = c(50, 100, 150))
```

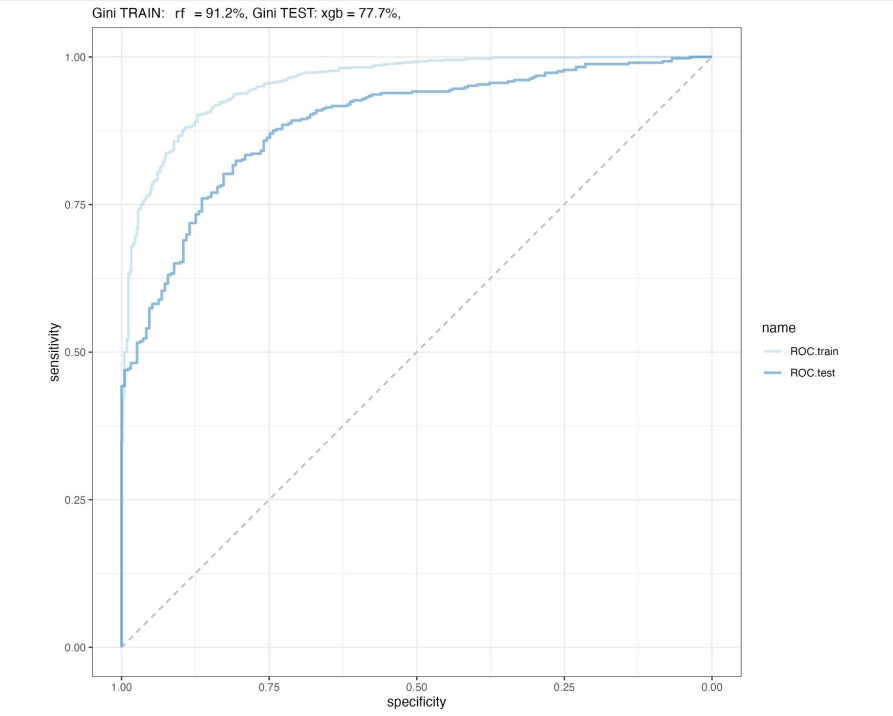


Results

iteration	gini_train	gini_test	models_spec	model_name
1	91.2	77.7	trees = 953, min_n = 75	random_forest
2	88.4	76.4	trees = 953, min_n = 100	random_forest
1	88.6	75.6	trees = 500, min_n = 25, tree_depth = 9, loss_reduction = 1, stop_iter = 10, learn_rate = 0.02	xgboost
2	88.6	75.6	trees = 500, min_n = 25, tree_depth = 9, loss_reduction = 1, stop_iter = 20, learn_rate = 0.02	xgboost
3	88.6	75.6	trees = 500, min_n = 25, tree_depth = 9, loss_reduction = 1, stop_iter = 30, learn_rate = 0.02	xgboost
3	84.7	74.9	trees = 953, min_n = 150	random_forest
2	77.1	61.2	cost_complexity = 0, tree_depth = 9, min_n = 25	decision_tree
1	72.3	57.8	cost_complexity = 0, tree_depth = 6, min_n = 25	decision_tree
3	66.1	53.1	cost_complexity = 0, tree_depth = 9, min_n = 50	decision_tree

ROC for selected models from table

random forest, iteration 1



xgb, iteration 2