Model Ideal

Code

Load Libraries

Hide

```
library(glmnet) # create model
library(caret) # ml library
library(tidyverse) # common libraries
```

Load Data

Hide

```
df<-read.csv("rent-amount-clear-preprocesing.csv")
```

Hide

```
attach(df)
```

Split Data

Hide

```
set.seed(2018) # random state

training.ids<-createDataPartition(rent.amount,p=0.7,list = F)

train_data<-df[training.ids,] # select train data index
test_data<-df[-training.ids,] # select test data index
```

Hide

```
X_train<- train_data %>% select(-rent.amount)
X_train<-as.matrix(X_train)# transform to matrix
Y_train<-train_data$rent.amount


X_test<- test_data %>% select(-rent.amount)
X_test<-as.matrix(X_test) # transform to matrix
Y_test<-test_data$rent.amount
```

Model Creation

Hide

```
lm_ridge<-glmnet(x=X_train,y=Y_train,alpha = 0.01,lambda = 0.001)
```

We assign the same parameters of the best model, which we had previously estimated.

Hide

```
predict_model<-function(data){

  y_pred<-predict(lm_ridge,newx =data)
  y_pred<-as.vector(y_pred)
}
```

Hide

```
pred_train<-predict_model(X_train)
pred_test<-predict_model(X_test)
```

Hide

```
train_data<- train_data %>% mutate(predictions=pred_train)

test_data<- test_data %>% mutate(predictions=pred_test)
```

MSE

Measures the average error between the predicted value and the original.

Train

Hide

```
train_data %>%

  summarise(mse=RMSE(predictions,rent.amount))
```

mse
<dbl>
0.1915118

1 row

Hide

```
NA
```

Test

Hide

```
test_data %>% summarise(mse=RMSE(predictions,rent.amount))
```

mse
<dbl>
0.2210363

1 row

R²

It measures the degree of fit of the model. The higher the value, the better the model fit.

Hide

```
train_data %>% summarise(mse=R2(predictions,rent.amount))
```

mse
<dbl>
0.9392662

1 row

Hide

```
test_data %>% summarise(mse=R2(predictions,rent.amount))
```

mse
<dbl>
0.9176224

1 row

Both metrics are very similar. Therefore, it indicates that our model is not overtrained. This means that the model is only good for the training data but it is unable to generalize with new data that it has never seen.

Exponential transformation

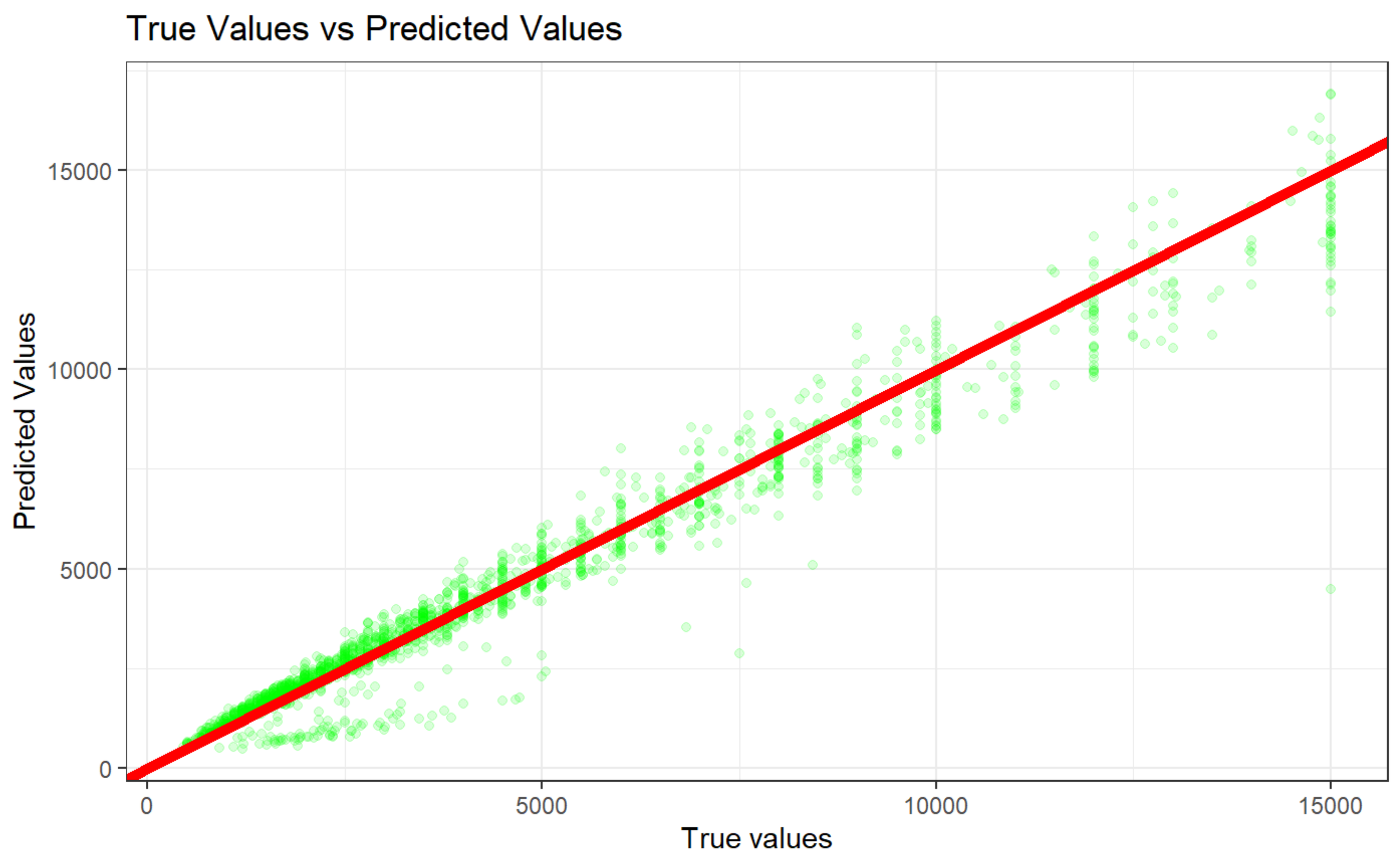
Hide

```
test_data<-test_data %>%

  mutate(rent.amount=exp(rent.amount)) %>%

  mutate(predictions=exp(predictions))
```

Let us remember that the rent.amount variable is in logarithmic scale and we perform the exponential transformation because it is its opposite operation.

Hide

```
test_data %>%
  ggplot(aes(x=rent.amount,y=predictions))+
  geom_point(color="green",alpha=0.15) +
  geom_abline(intercept = 0,slope = 1,color="red",lwd=2) +
  theme_bw() +
  labs(title = "True Values vs Predicted Values",
       x="True values",y="Predicted Values")
```



Hide

```
test_data %>% select(rent.amount,predictions) %>% head(50)
```

	rent.amount	predictions
	<dbl>	<dbl>
7	5000	4578.1800
10	2900	2905.8427
11	720	833.7263
15	3950	4048.1007
17	3010	3235.8558
18	3500	3670.0866
19	7800	7240.4806
23	3500	3564.3342
27	6500	5467.9499
29	1800	1881.6973

1-10 of 50 rows Previous 1 2 3 4 5 Next

In most predictions, it gives predictions very close to the original value.

Save Model

Hide

```
saveRDS(lm_ridge,"lm_ridge_rent.RDS")
```