Lab06

Amanda Mentesana

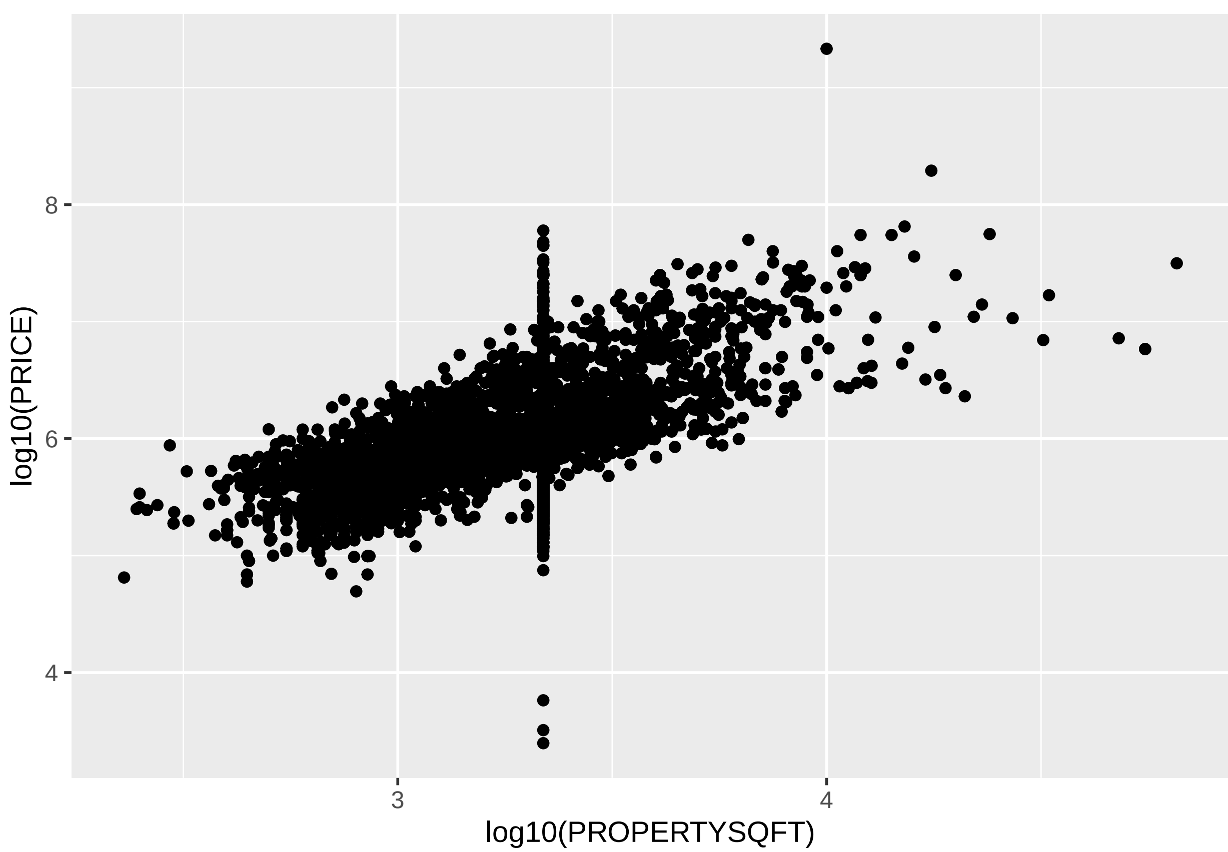
2025-03-28

Train and Evaluate 3 Regression Models Predicting Price from Square Footage using MAE, MSE, and RMSE

1) Linear Regression Model

```
## 1) Untrained Model

## Plot dataset to identify best shape and potential outliers
ggplot(ny_housing, aes(x = log10(PROPERTYSQFT), y = log10(PRICE))) + geom_point() #this is a more unders
```



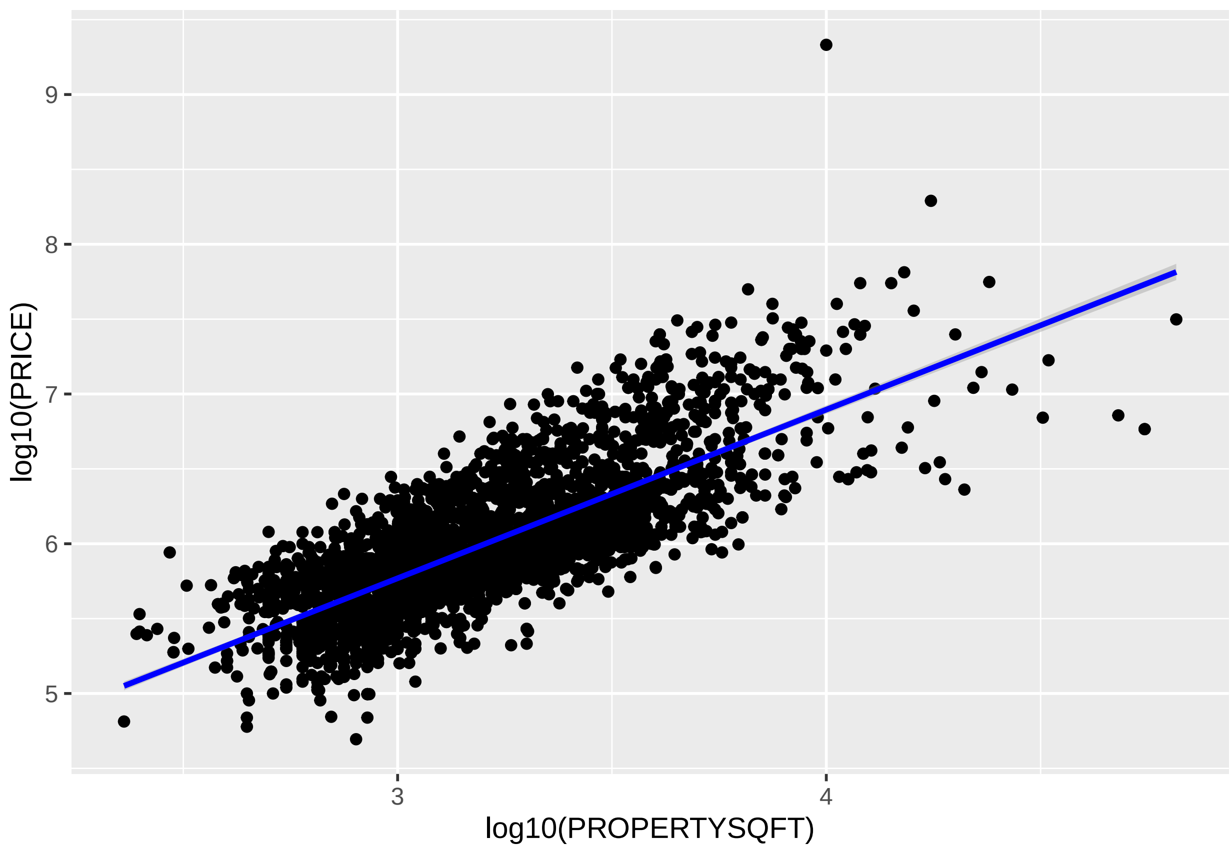
```r
# Clean dataset for weird repeating outlier value.
ny_housing <- ny_housing[-which(ny_housing$PROPERTYSQFT==2184.207862),]

lin.mod <- lm(log10(PRICE) ~ log10(PROPERTYSQFT), ny_housing)
summary(lin.mod) #multiple R-squared of 0.5828
```

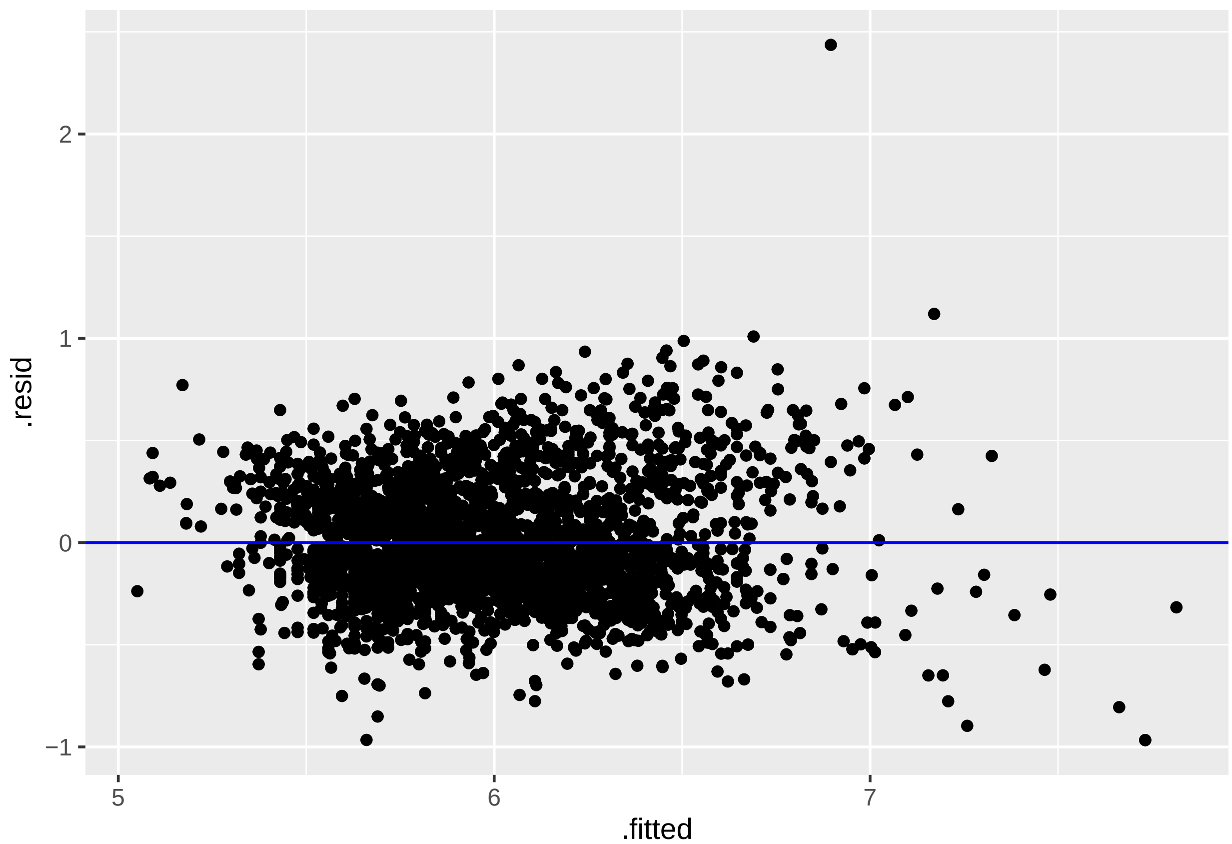
```
##
## Call:
## lm(formula = log10(PRICE) ~ log10(PROPERTYSQFT), data = ny_housing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9666 -0.1999 -0.0506  0.1920  2.4363
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.39126    0.05443   43.93   <2e-16 ***
## log10(PROPERTYSQFT)  1.12609    0.01690   66.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2926 on 3178 degrees of freedom
## Multiple R-squared:  0.5828, Adjusted R-squared:  0.5827
## F-statistic:  4440 on 1 and 3178 DF,  p-value: < 2.2e-16
```

```r
# Plot cleaned dataset and linear regression fit
ggplot(ny_housing, aes(x = log10(PROPERTYSQFT), y = log10(PRICE))) + geom_point() + stat_smooth(method =
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(lin.mod, aes(x = .fitted, y = .resid)) + geom_point() + geom_hline(yintercept = 0, col="blue") #
```




```
## Trained Model

## split train/test, these indicies will be used for testing other models as well.
train.indexes <- sample(nrow(ny_housing),0.7*nrow(ny_housing))
train <- ny_housing[train.indexes,]
test <- ny_housing[-train.indexes,]

## LM train
lin.mod.train <- lm(log10(PRICE) ~ log10(PROPERTYSQFT), train)

summary(cv(lin.mod))
```

```
## R RNG seed set to 870550
```

```
## 10-Fold Cross Validation
## method: Woodbury
## criterion: mse
## cross-validation criterion = 0.08569959
## bias-adjusted cross-validation criterion = 0.08569116
## 95% CI for bias-adjusted CV criterion = (0.08007893, 0.09130339)
## full-sample criterion = 0.08553966
```

```
lm.pred <- predict(lin.mod, test)

## err = predicted - real
```

```r
err <- lm.pred-log10(test$PRICE)

## MAE
abs.err <- abs(err)
mean.abs.err <- mean(abs.err)

## MSE
sq.err <- err^2
mean.sq.err <- mean(sq.err)

## RMSE
sq.err <- err^2
mean.sq.err <- mean(sq.err)
root.mean.sq.err <- sqrt(mean.sq.err)


lin.mod.train.df <- data.frame(mean.abs.err, mean.sq.err, root.mean.sq.err)
lin.mod.train.df
```

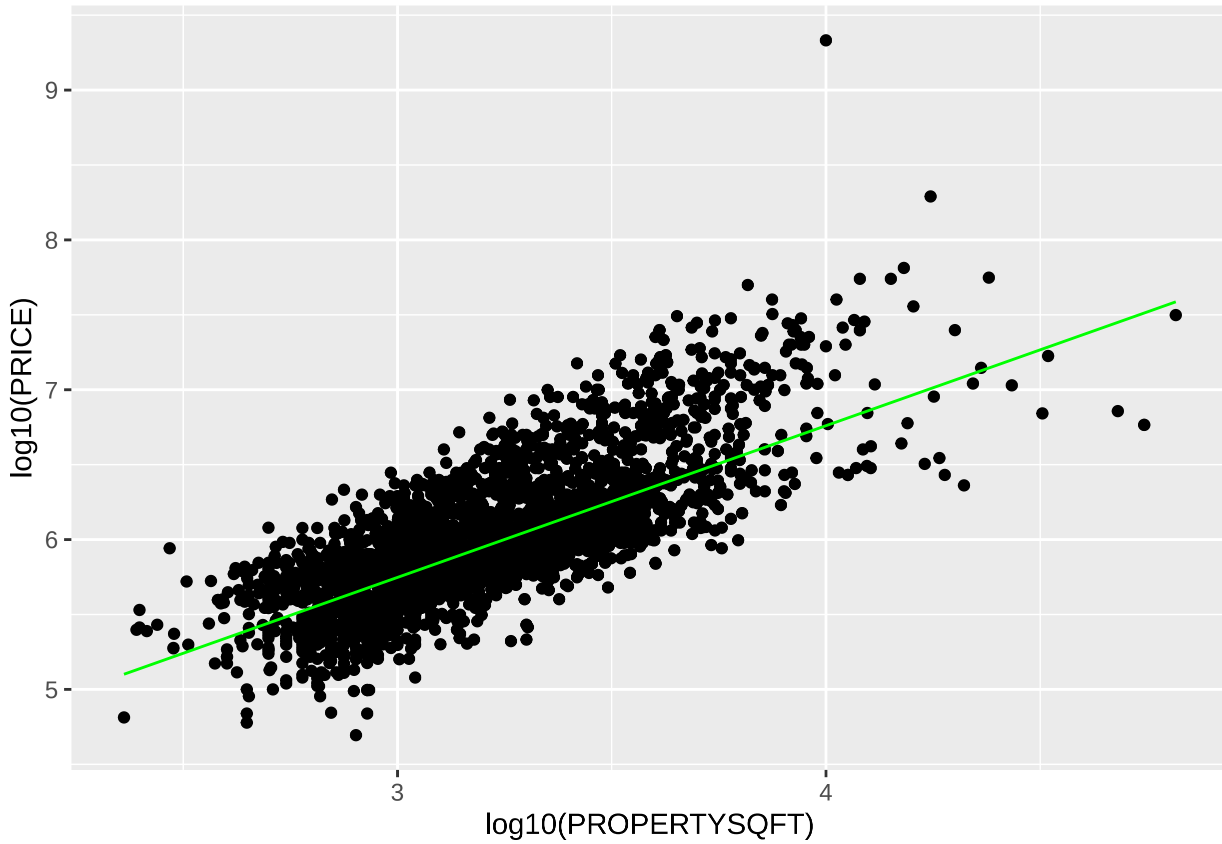
```
##   mean.abs.err mean.sq.err root.mean.sq.err
## 1    0.2313867  0.08307122        0.2882208
```

2) SVM- Linear Model

```r
svm.lin.mod0 <- svm(log10(PRICE) ~ log10(PROPERTYSQFT), ny_housing, kernel="linear")

svm.lin.pred0 <- predict(svm.lin.mod0, ny_housing)

ggplot(ny_housing, aes(x = log10(PROPERTYSQFT), y = log10(PRICE))) +
  geom_point() +
  geom_line(aes(x=log10(PROPERTYSQFT), y=svm.lin.pred0), col="green")
```




```
## Train the SVM Linear Model
## Linear SVM Model
k = 100
mae <- c()
mse <- c()
rmse <- c()

for (i in 1:k) {
  train.indexes <- sample(nrow(ny_housing),0.7*nrow(ny_housing))

  train <- ny_housing[train.indexes,]
  test <- ny_housing[-train.indexes,]

  svm.lin.mod <- svm(log10(PRICE) ~ log10(PROPERTYSQFT), ny_housing, kernel="linear")

  svm.lin.pred <- predict(svm.lin.mod, test)

  err <- svm.lin.pred-log10(test$PRICE)

  abs.err <- abs(err)
  mean.abs.err <- mean(abs.err)

  sq.err <- err^2
  mean.sq.err <- mean(sq.err)

  root.mean.sq.err <- sqrt(mean.sq.err)
```

```
  mae <- c(mae, mean.abs.err)
  mse <- c(mse, mean.sq.err)
  rmse <- c(rmse, root.mean.sq.err)
}

mae.m <- c()
mse.m <- c()
rmse.m <- c()
mae.m <- mean(mae)
mse.m <- mean(mse)
rmse.m <- mean(rmse)
svm.lin.df <- data.frame(mae.m, mse.m, rmse.m)
svm.lin.df
```

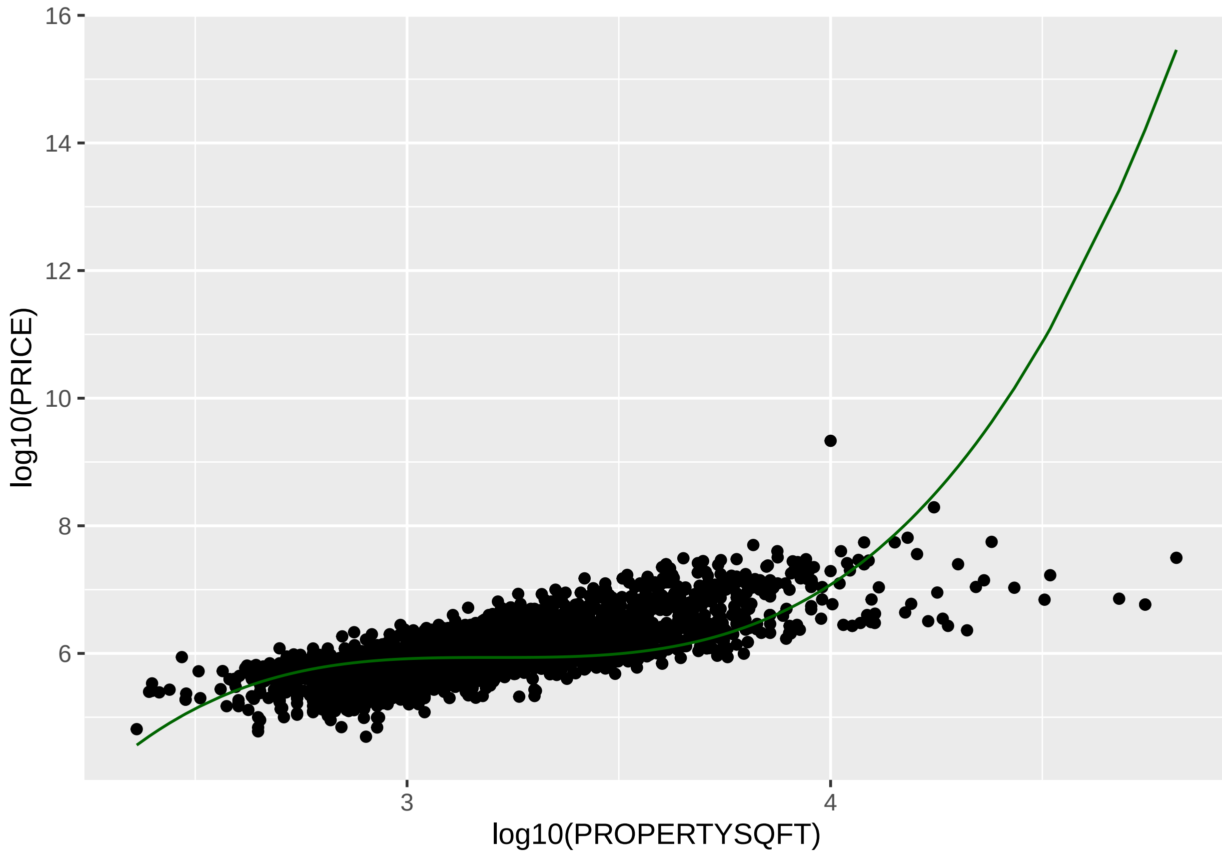
```
##      mae.m      mse.m    rmse.m
## 1 0.229249 0.08953817 0.2991028
```

3) SVM- Polynomial Model

```
# Untrained model plotted

svm.poly.mod0 <- svm(log10(PRICE) ~ log10(PROPERTYSQFT), ny_housing, kernel="polynomial")
svm.poly.pred0 <- predict(svm.poly.mod0, ny_housing)

ggplot(ny_housing, aes(x = log10(PROPERTYSQFT), y = log10(PRICE))) +
  geom_point() +
  geom_line(aes(x=log10(PROPERTYSQFT), y=svm.poly.pred0), col="darkgreen") #this plot likely is not the
```




```r
## Polynomial SVM Model
k = 100
mae <- c()
mse <- c()
rmse <- c()

for (i in 1:k) {
  train.indexes <- sample(nrow(ny_housing),0.7*nrow(ny_housing))

  train <- ny_housing[train.indexes,]
  test <- ny_housing[-train.indexes,]

  svm.pol.mod <- svm(log10(PRICE) ~ log10(PROPERTYSQFT), ny_housing, kernel="polynomial")

  svm.pol.pred <- predict(svm.pol.mod, test)

  err <- svm.pol.pred-log10(test$PRICE)

  abs.err <- abs(err)
  mean.abs.err <- mean(abs.err)

  sq.err <- err^2
  mean.sq.err <- mean(sq.err)

  root.mean.sq.err <- sqrt(mean.sq.err)
```

```r
  mae <- c(mae,mean.abs.err)
  mse <- c(mse,mean.sq.err)
  rmse <- c(rmse,root.mean.sq.err)
}

mae.m <- c()
mse.m <- c()
rmse.m <- c()
mae.m <- mean(mae)
mse.m <- mean(mse)
rmse.m <- mean(rmse)
svm.pol.df <- data.frame(mae.m, mse.m, rmse.m)
svm.pol.df
```

```
##      mae.m     mse.m    rmse.m
## 1 0.285453 0.2255915 0.4720561
```

```r
# Comparison of error of the three models

lin.mod.train.df
```

```
##   mean.abs.err mean.sq.err root.mean.sq.err
## 1    0.2313867  0.08307122        0.2882208
```

```r
svm.lin.df
```

```
##      mae.m      mse.m    rmse.m
## 1 0.229249 0.08953817 0.2991028
```

```r
svm.pol.df ## Even though the polynomial model was the least helpful in terms of being specific to the
```

```
##      mae.m     mse.m    rmse.m
## 1 0.285453 0.2255915 0.4720561
```