Property prices in New York

James Edholm

19 October 2020

Introduction

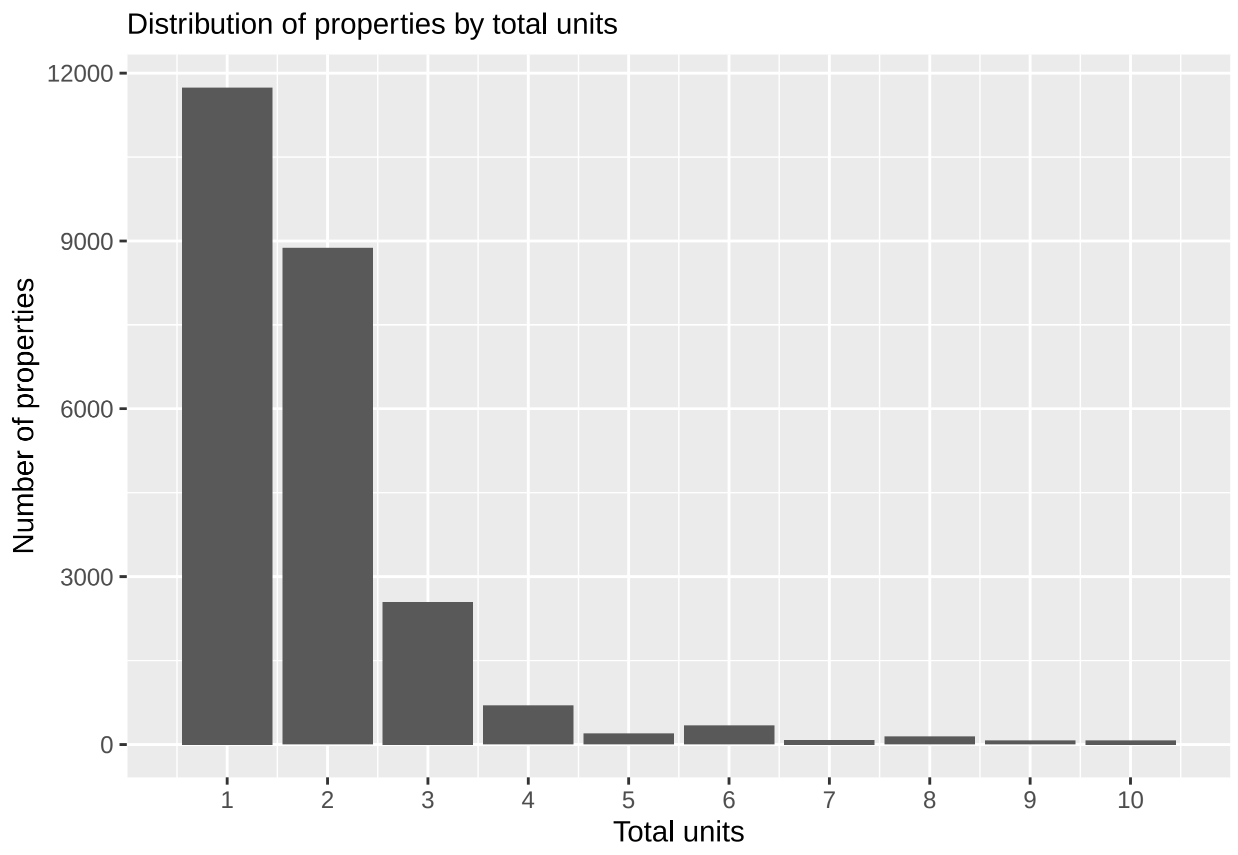
We will use the machine learning packages of R to investigate if we can predict the sale prices of properties in New York, based on characteristics such as what borough the property is in, the amount of land it has, the number of total units in the property, the square feet of area inside the property, its tax class and the decade it was built.

Overview of data set

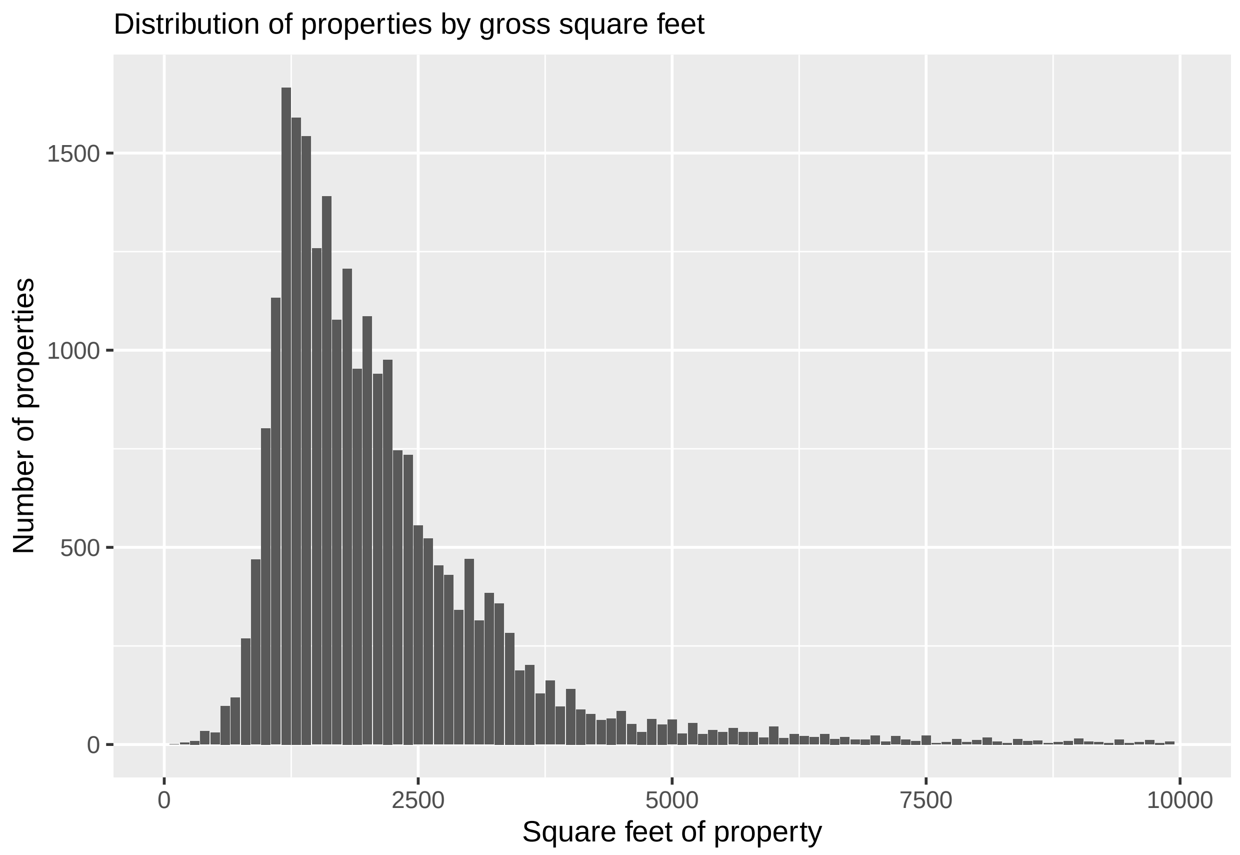
After removing entries which are simple transfers of deeds rather than actual sales, the data set contains 25,202 property sales along with the borough, neighbourhood, building class category, current tax code, block, lot, easement, current building class, address, appartment number, zip code, number of residential units, number of commercial units", total units, square feet of land, gross square feet, year built, tax class at time of sale, building class at time of sale and the sale date.

We can make several plots of the dataset, showing how the properties are distributed by borough, gross square feet, square feet of land and decade built.

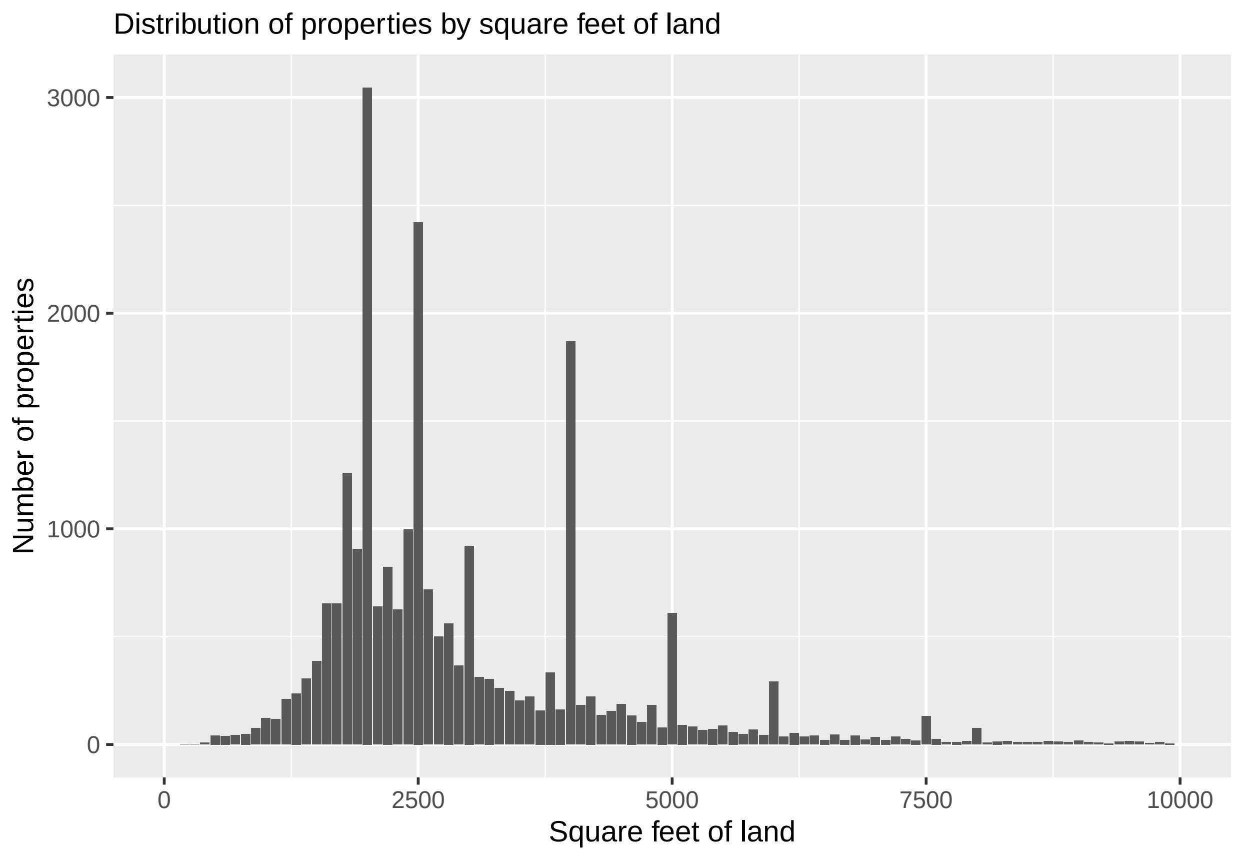
```r
#plot number of sales by total units
prices %>%
  group_by(Total_units) %>%
  summarise(number= n()) %>%
  ggplot(aes(x=Total_units, y=number)) +
  geom_bar(stat="identity")+
  scale_x_continuous(breaks=c(1,2,3,4,5,6,7,8,9,10), limits=c(0.5,10.5))+
  labs(x="Total units", y= "Number of properties",
       subtitle="Distribution of properties by total units")
```



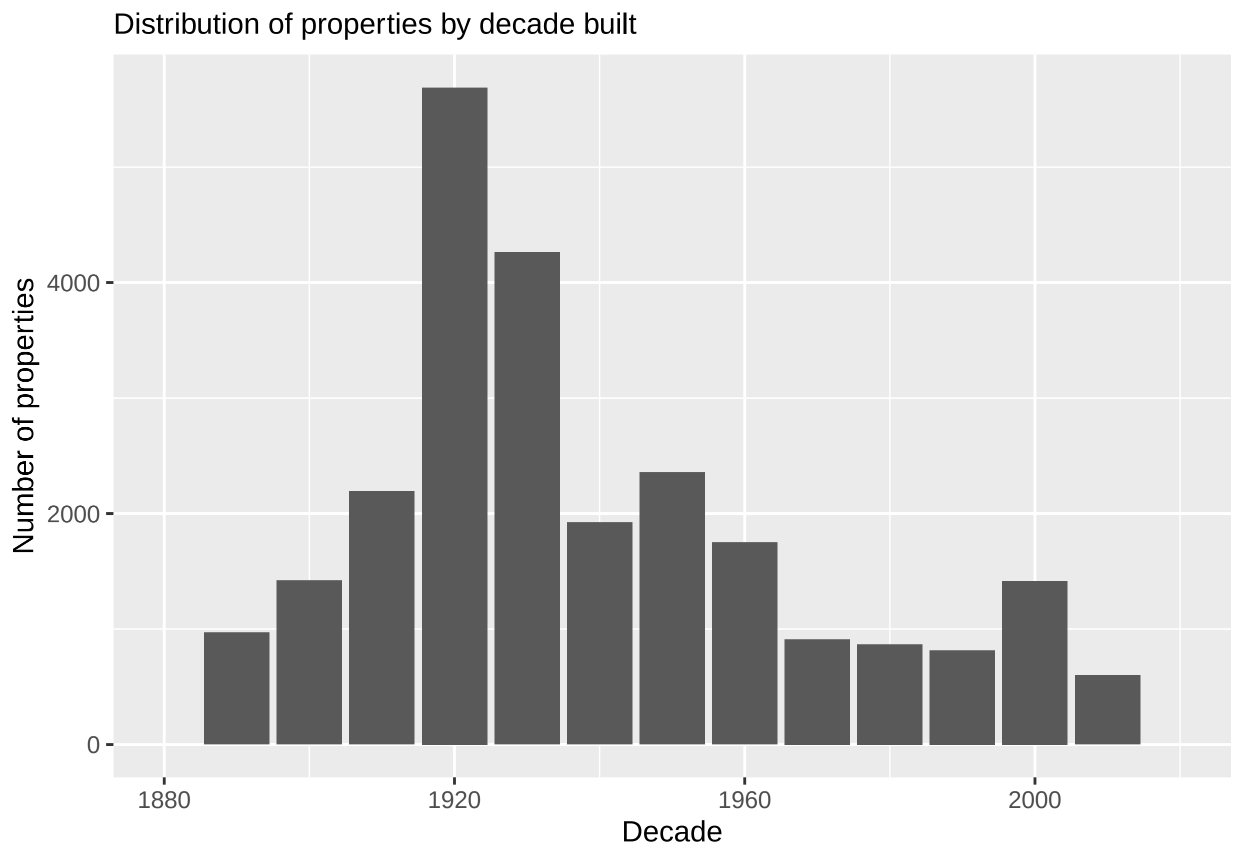
```
#plot number of sales by  gross square feet
prices %>%
  group_by(GSF) %>%
  summarise(number= n()) %>%
  ggplot(aes(x=GSF, y=number)) +
  geom_bar(stat="identity")+
  xlim(0,10000)+
  labs(x="Square feet of property", y= "Number of properties",
       subtitle= "Distribution of properties by gross square feet")
```



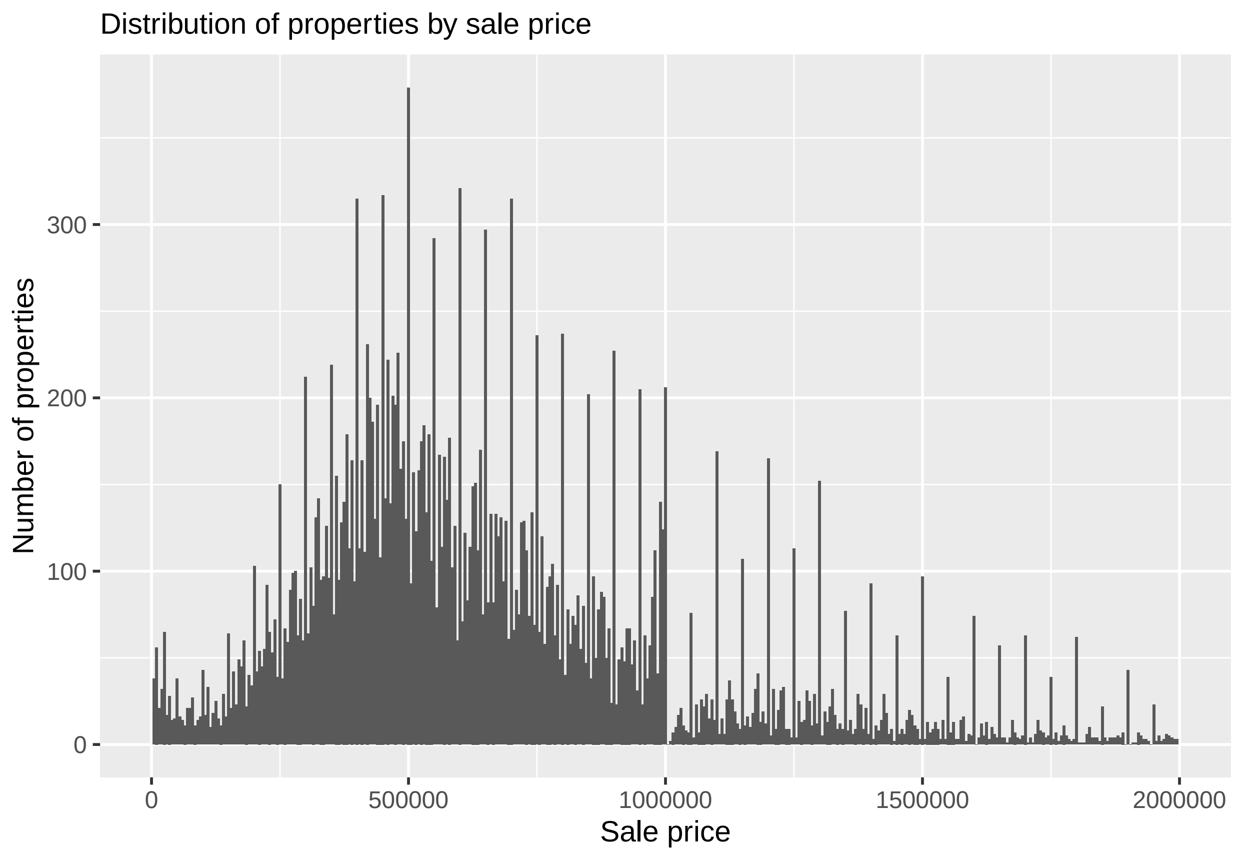
```
#plot number of sales by square feet of land
prices %>%
  group_by(LSF) %>%
  summarise(number= n()) %>%
  ggplot(aes(x=LSF, y=number)) +
  geom_bar(stat="identity") +
  xlim(0,10000)+
  labs(x="Square feet of land", y= "Number of properties",
       subtitle= "Distribution of properties by square feet of land")
```



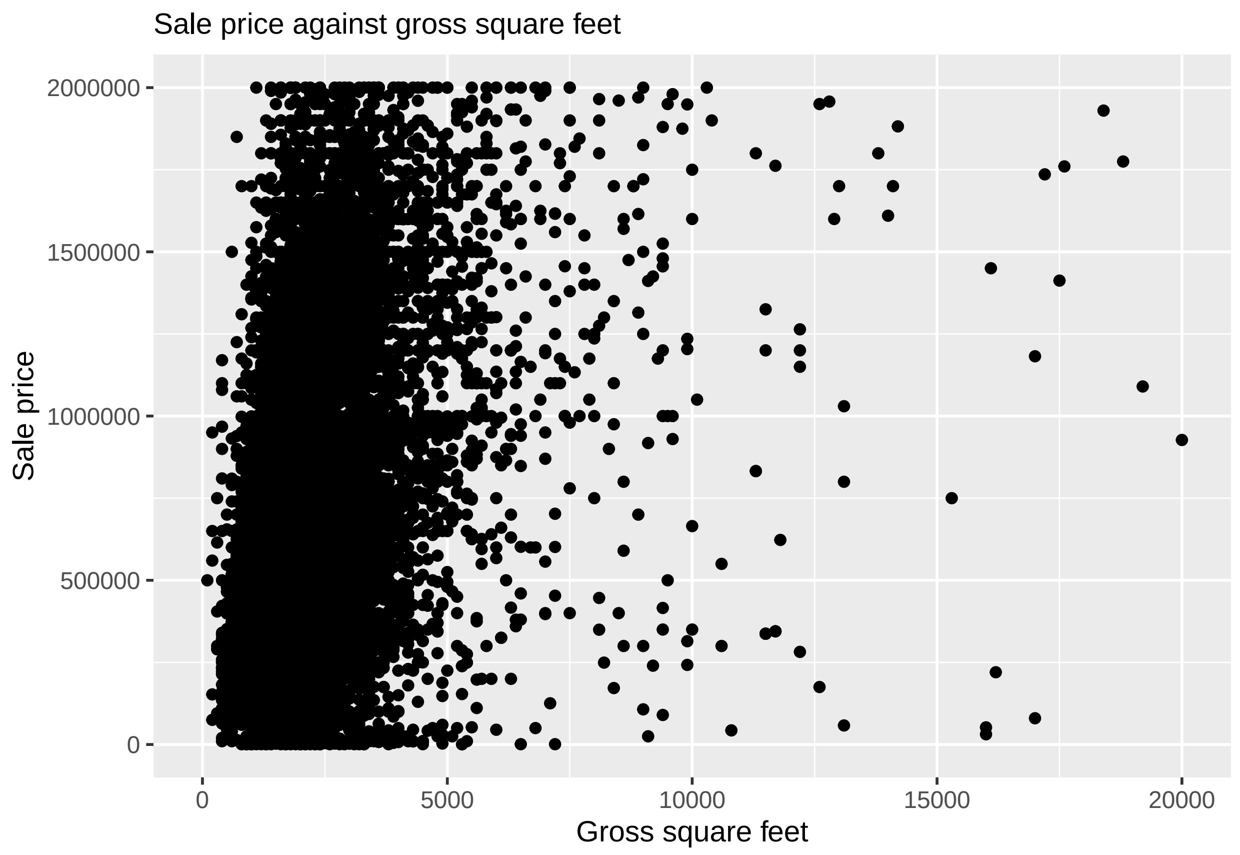
```
#plot number of sales by decade
prices %>%
  group_by(Decade) %>%
  summarise(number= n()) %>%
  ggplot(aes(x=Decade, y=number)) +
  geom_bar(stat="identity") +
  scale_x_continuous(limits=c(1880,2020)) +
  labs(x="Decade", y= "Number of properties",
       subtitle="Distribution of properties by decade built")
```



```
#plot a histogram of sale prices
prices %>%
  ggplot(aes(Sale_price)) +
  geom_histogram(binwidth=5000)+
  xlim(0,2*10^6)+
  labs(x="Sale price", y="Number of properties",
       subtitle= "Distribution of properties by sale price")
```



```
prices %>%
  ggplot(aes(x=GSF, y=Sale_price)) +
  geom_point()+
  xlim(0,2*10^4)+
  ylim(0,2*10^6)+
  labs(x="Gross square feet", y="Sale price",
       subtitle= "Sale price against gross square feet")
```



Methods

Introduction to method

We use the machine learning packages to investigate if we can predict the sale price of a property, given the borough, gross square feet, square feet of land and the decade built. Initially, we expected that prices would rice if the gross square footage and square feet of land increases, but how the price depends on the borough or decade built is less obvious. We surprisingly found however that the borough, the amount of land with the property and the decade the property was built in were not good predictors of the sale price.

Building the model

We will investigate the Root Mean Square Error (RMSE) given by different methods. We want to minimise the RMSE, which represents the difference between the predicted price and the actual price. We created the prices data set and the validation data set. The validation data set contains 2,782 ratings (i.e. about 10% of the prices data set).

I started by splitting the prices data into training data (train_set) and test data (test_set). I then tried several models in order to find the best method. Unless otherwise stated, the RMSE is for predictions for the test data *test_set* based on the training data *test_set*.

```
options(warn = oldw)
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_i <- createDataPartition(y = prices$Sale_price, times = 1, p = 0.3, list = FALSE)
train_set <- prices[-test_i,]
test_set_temp <- prices[test_i,]
test_set <- test_set_temp %>%
  semi_join(train_set, by = "Total_units") %>%
  semi_join(train_set, by = "GSF") %>%
  semi_join(train_set, by = "Building_class_sale")
```

Only including the overall average

```
#predict purely based on mean of total data set
mu <- mean(train_set$Sale_price)
mu
```

```
## [1] 928436.7
```

```
predicted_price <- test_set %>%
  mutate(pred = mu ) %>%
  pull(pred) #extract the predictions as a list

#test how good this prediction is
rmse_mu <- RMSE(predicted_price, test_set$Sale_price)
rmse_mu
```

```
## [1] 1013723
```

As a starting point, we can predict simply that all sales had a price which the average of the training data μ, which was $928k. This gave an error of $1 million compared to the actual price. This is the RMSE, the average error compared to the true value.

Corrections for total units

```
#add a term b_b which adjusts predicted price based on borough
unit_avgs <- train_set %>%
  group_by(Total_units) %>% #group by borough
  summarize(b_u = mean(Sale_price - mu))

#produce predictions based on adjusting depending on total units
predicted_price <- test_set %>%
  left_join(unit_avgs, by='Total_units') %>%
  mutate(pred = mu + b_u) %>%
  pull(pred)

#test how good this prediction is
rmse_units <- RMSE(predicted_price, test_set$Sale_price)
rmse_units
```

```
## [1] 892482.6
```

Next, we include a correction for total units, such that if the average price for properties with a certain number of units was $1 million, the model would predict a price of $1 million for any property with that many total units. This generated an RMSE of $892k.

Corrections for gross square footage

```r
#Include a term b_g which adjusts price based on gross square feet
sqft_avgs <- train_set %>%
  left_join(unit_avgs, by='Total_units') %>%
  group_by(GSF) %>%       #group by user
  summarize(b_g = mean(Sale_price - mu - b_u))

#Predict price based on borough and square footage of property
predicted_price <- test_set %>%
  left_join(unit_avgs, by='Total_units') %>%
  left_join(sqft_avgs, by='GSF') %>%
  mutate(pred = mu + b_u + b_g) %>%
  pull(pred)

NAs_pred <- which(is.na(predicted_price))
predicted_price[NAs_pred] <- mu
#test how good this prediction is
rmse_sqft <- RMSE(predicted_price, test_set$Sale_price)
rmse_sqft
```

```
## [1] 889442
```

We introduce a further correction by adding a term which corrects for square footage. If houses with a certain square footage had a lower than expected sale price given the number of total units, then the model would predict a lower sale price for other properties with that square footage. I calculated the average difference from the expected price, $b_g = \text{Sale price} - \mu - b_u$, and added that to the prediction. Using this more sophiscated method gave an RMSE of $889k.

Correcting for building class

```r
#Include a term based on building class
bclass_avgs <- train_set %>%
  left_join(unit_avgs, by='Total_units') %>%
  left_join(sqft_avgs, by='GSF') %>%
  group_by(Building_class_sale) %>%
  summarize(b_c = mean(Sale_price - mu - b_u - b_g ))

#Predict price based on borough, square footage, square feet of land,
#building class
predicted_price <- test_set %>%
  left_join(unit_avgs, by='Total_units') %>%
  left_join(sqft_avgs, by='GSF') %>%
  left_join(bclass_avgs, by='Building_class_sale') %>%
  mutate(pred=ifelse(is.na(b_c), mu + b_u + b_g,
                     mu + b_u + b_g + b_c)) %>%
  pull(pred)

#test how good this prediction is
```

```
rmse_class <- RMSE(predicted_price, test_set$Sale_price)
rmse_class
```

```
## [1] 845621.4
```

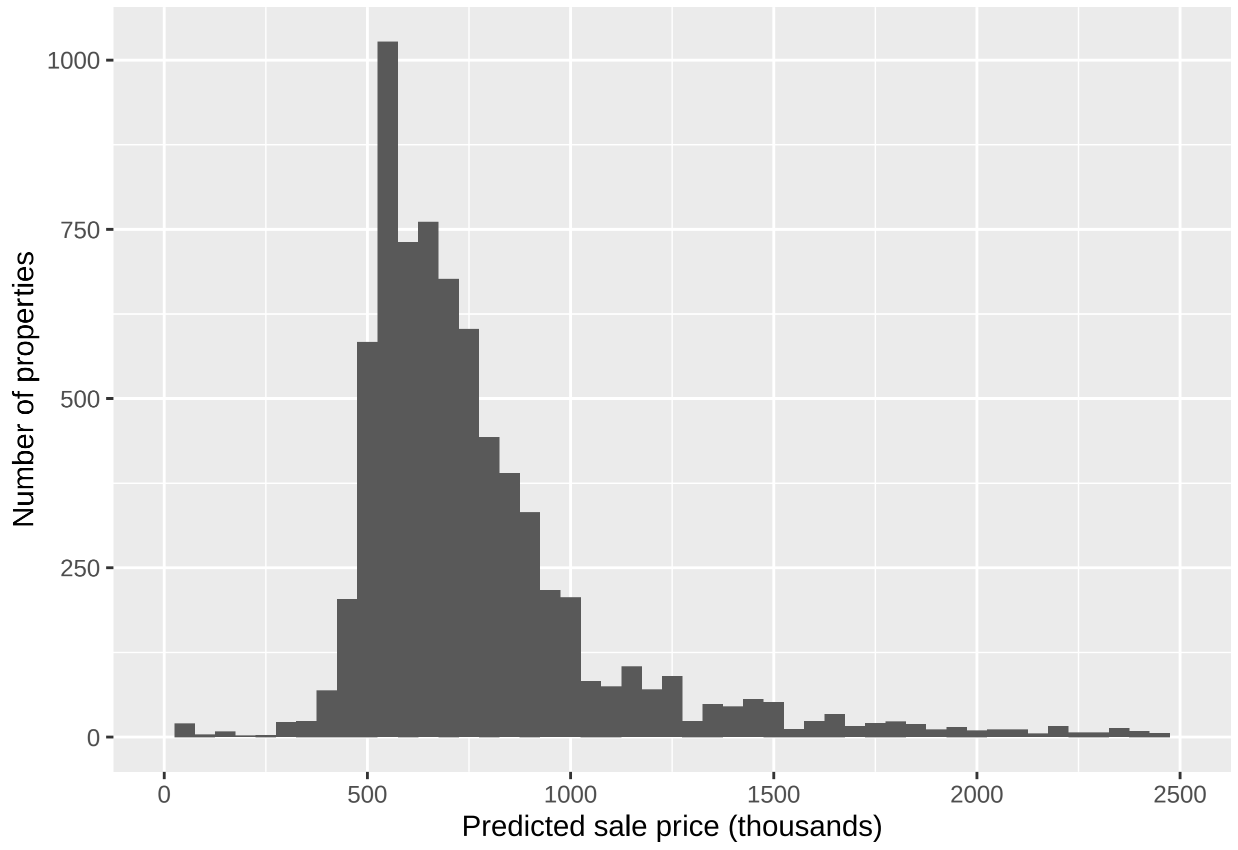
Finally, we add a term to adjust for building class. This generates an RMSE of $846k.

We now plot the distribution of our predicted prices, which looks reasonably similar to the actual distribution.

```
dataf=data.frame(predicted_price)
ggplot(dataf, aes(x=predicted_price/10^3)) +
  geom_histogram(binwidth=50) +
  labs(x="Predicted sale price (thousands)", y="Number of properties")+
  xlim(0,2.5*10^3)
```

```
## Warning: Removed 232 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



Regularisation

We trained our method on sales with a wide variety of entries, including some classes that only had one entry. Estimates for these properties should not be trusted, because the low numbers of entries increases the uncertainty.

Rather than just taking these estimates, it is normally better to make a prediction that is closer to the mean of all the sales. Regularisation is the process of penalising correction terms that are generated by a low number of entries (low number of sales in this case).

The penalty is given by adding an extra term to the sum of squares, which punishes predictions which are both far away from the mean and based on a small amount of data. In order to minimise this new equation, we use a new b_i, given by

$\hat{b}_i = \frac{\sum(\text{Sale price}-\mu)}{n_i+\lambda}$

where λ is a tuning parameter and n_i is the number of sales in the category i.

Regularisation method

```r
#split training data into two sets
test_i <- createDataPartition(y = train_set$Sale_price, times = 1, p = 0.3, list = FALSE)
train_set_1 <- train_set[-test_i,]
train_set_2 <- train_set[test_i,]
train_set_2 <- train_set_2 %>%
  semi_join(prices, by = "Total_units") %>%
  semi_join(prices, by = "GSF")

#find a rough estimate for value of lambda which minimises RMSE
lambdas <- seq(0, 20, 1)
rmses <- sapply(lambdas, rmse_lambda, train= train_set_1, test=train_set_2)
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

```r
#Find a more precise value of lambda which minimises RMSE
lambdas_narrow <- seq(lambda-2, lambda+2, 0.25)
rmses_narrow <- sapply(lambdas_narrow, rmse_lambda, train= train_set_1, test=train_set_2)
lambda <- lambdas_narrow[which.min(rmses_narrow)]
lambda
```

```
## [1] 5.25
```

```r
#Test model on test set
reg_rmse <- rmse_lambda(lambda, train_set, test_set)
```

We split the training data up into two sets, with training_set_1 representing 70% of the total, then tested which value of λ minimised the RMSE (by training on the first part of the training set with several values of λ and then testing on the second part). We started by taking values of λ between 0 and 10, with gaps of 0.25. This showed that the RMSE was minimised when $\lambda = 5.25$.

We then tested our model with this value of λ on the test set, which gave an RMSE of $800k.

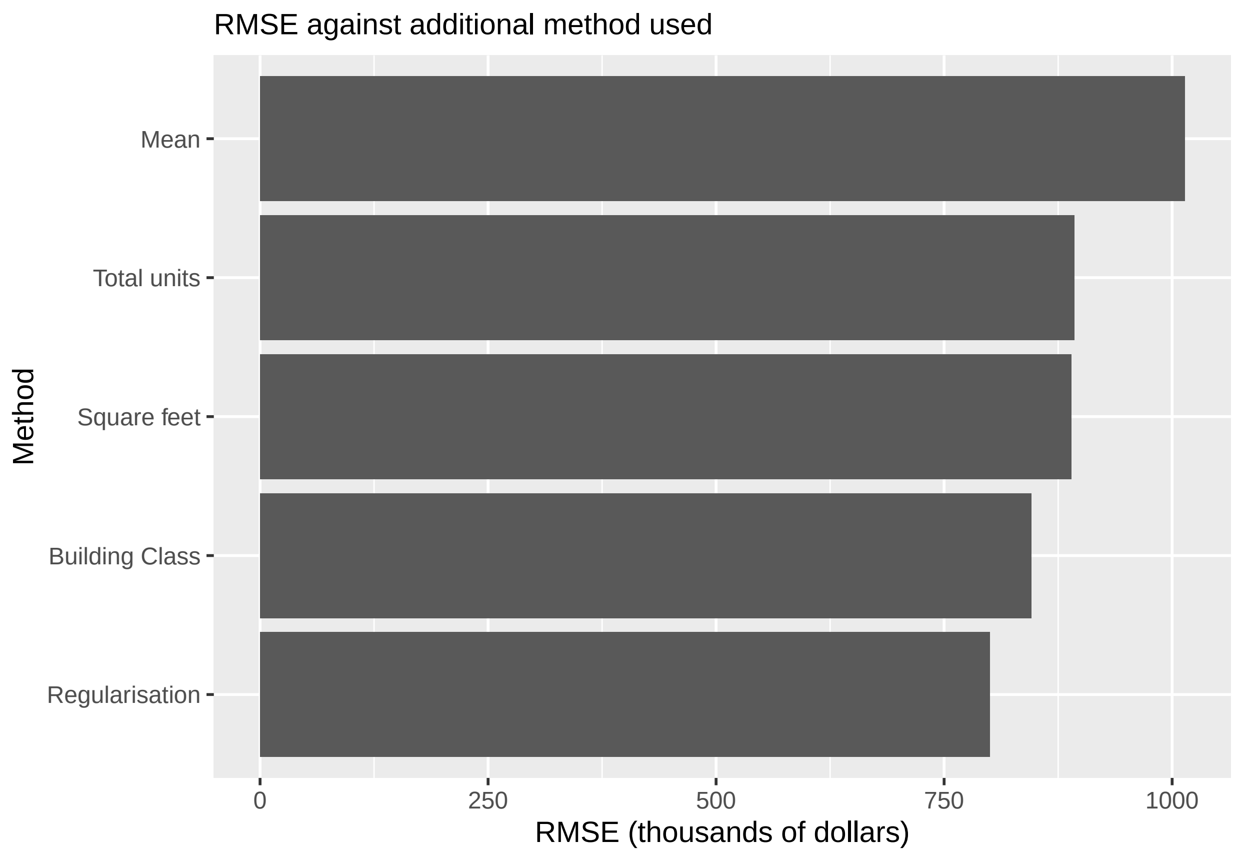
Results

To predict the price of a certain property, we started with the mean of the training data. For our final model, we corrected for the total units, the gross square footage and finally the building class at sale.

The corrections for total units and building class produced the most improvement, and correcting for gross square feet produced only a small improvement. The final improvement to my model was to include regularisation, which produced a significant improvement.

I show how each additional method improved the RMSE in the following bar charts.

```
names_rmse <- c("Mean", "Total units", "Square feet", "Building Class",
                "Regularisation")
results_rmse <- c(rmse_mu/10^3, rmse_units/10^3, rmse_sqft/10^3, rmse_class/10^3, reg_rmse/10^3)
df <- data.frame(names_rmse, results_rmse)
ggplot(data=df, aes(x= reorder(names_rmse,  results_rmse), y=results_rmse)) +
  geom_bar(stat="identity") +
  coord_flip() +
  labs(x="Method", y="RMSE (thousands of dollars)",
       subtitle="RMSE against additional method used")
```



```
#Find RMSE of model on validation data set
val_rmse <- rmse_lambda(lambda, prices, validation)
```

Finally we tested our method on the validation set, resulting in an RMSE of $766k.

Conclusion

We produced a model which predicts what price a property will sell for, based on the number of units, square footage and building class. We then added a correction whereby predicted sales which are calculated based on a low number of sales are moved closer to the mean (regularisation). This method produced an RMSE of $766k.

We surprisingly found that the amount of land with the property and the decade the property was built in were not good predictors of the sale price.

The limitations of the model are that it would only work for properties where there are already existing sales with the same number of total units, square feet (to the nearest hundred) and the same building class.

Future work could include varying the value of λ used to regularise the model so that there is a different value of λ for different adjustments (e.g. total units or building class). We could also investigate the specific interaction between different characteristics, e.g. perhaps square footage has a small effect on prices for houses, but a large effect on prices for apartments.