KEY MACHINE LEARNING CONCEPTS – EXPLAINED WITH LINEAR REGRESSION

This PDF version is not interactive. For interactivity, use the html version instead

Carsten Lange
clange@cpp.edu
Cal Poly, Pomona

Textbook

LOADING DATA AND LIBRARIES

WHAT WILL YOU LEARN

- Reviewing the basic idea behind linear regression
- Learning how how to measure predictive quality with Mean Squared Error (MSE).
- Understanding the role of parameters in linear regression and in machine learning models
- Calculating optimal regression parameters using OLS
- Finding optimal regression parameters by trial and error
- Distinguish between unfitted and fitted models
- Using the `tidymodels` package to split observations from a dataset randomly into a training and testing dataset.
- Understanding how categorical data such as the sex of a person (female/male) can be transformed into numerical dummy variable.
- Being able to distinguish between dummy encoding and one-hot encoding
- Using `tidymodels` including model design and data pre-processing (recipes) to analyze housing prices.

JUMPING RIGHT INTO IT

UNIVARIATE OLS WITH A REAL WORLD DATASET

Data Description:

- King County House Sale dataset (Kaggle 2015). House sales prices from May 2014 to May 2015 for King County in Washington State.
- Several predictor variables:
  For now we use only $Sqft$
- For simplicity, we use only 100 randomly chosen observations from the total of 21,613 observations.

LOADING THE DATA AND ASSIGNING TRAINING AND TESTING DATA

Note: To ensure compatibility with a JavaScript simulation used in the following slides, as an exception, *training* and *testing data* are generated with `n_sample()` rather than with `tidymodels` commands.

```
DataTrain:

   Price Sqft
1 517000 1180
2 236000 1300
3 490000 2800
4 129000 1150
5 257000 1400
6 312500  870

DataTest:

   Price Sqft
1 743000 2110
2 175000  620
3 292000 2270
4 775000 3020
5 305495 2110
6 265000  960
```

HOW MUCH IS A HOUSE WORTH IN KING COUNTY?

A house with average properties should be predicted with an average price!

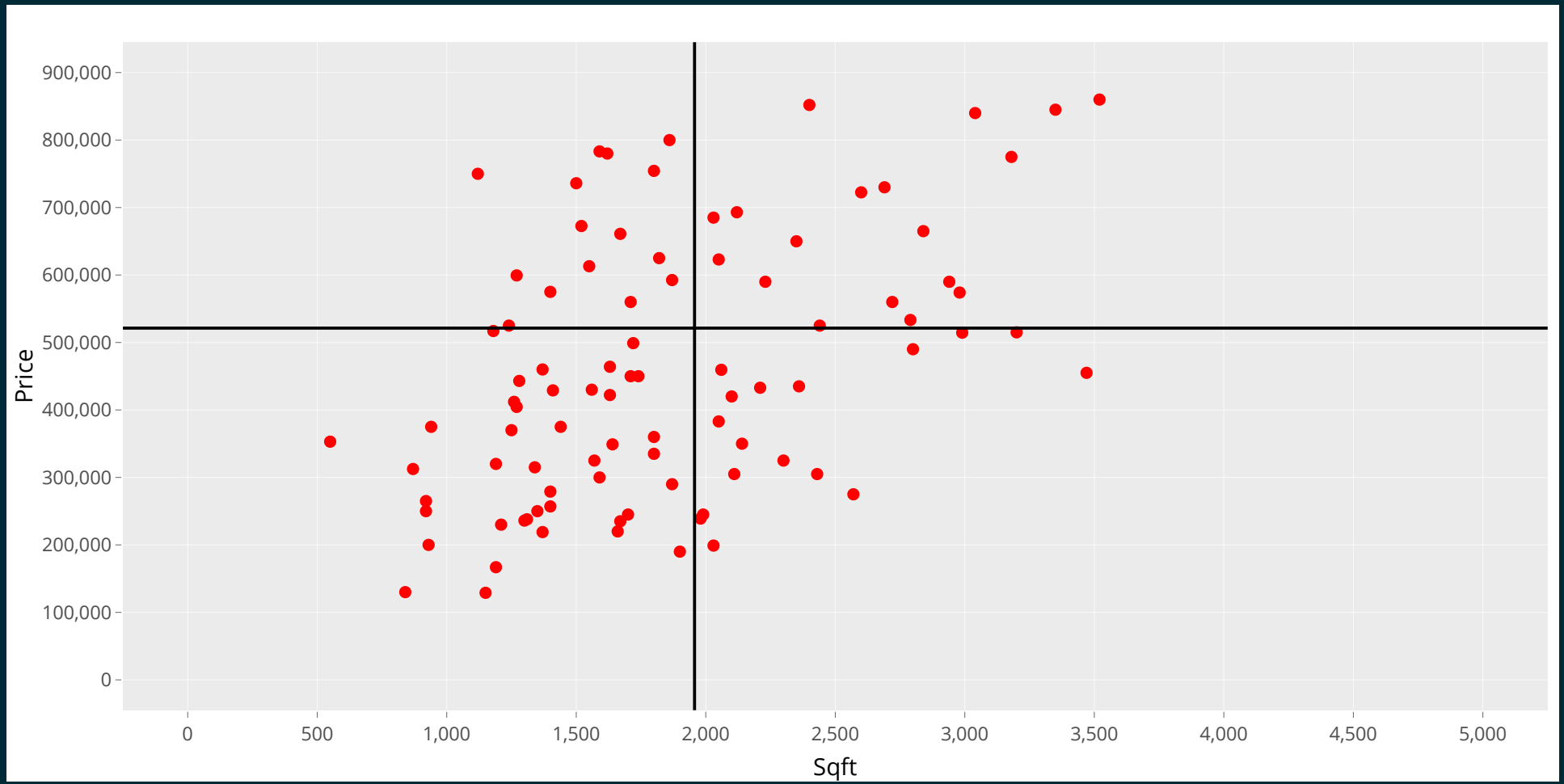
▶ Code

```
The mean square footage of a house in King county is: 1956.7
```

▶ Code

```
The mean price of a house in King county is: 521294.2
```

PREDICTING THE PRICE OF AN AVERAGE SIZED HOUSE AS THE AVERAGE OF ALL HOUSE PRICES



Textbook

AN INTERACTIVE GRAPH THAT EXPLAINS IT ALL

Unfitted Linear Regression Model:

$$\underbrace{\widehat{Price}}_{\hat{y}} = \underbrace{\beta_1}_{m}\underbrace{Sqft}_{x} + \underbrace{\beta_2}_{b}$$

JavaScript Simulation for Fitting Parameters:
https://econ.lange-analytics.com/calcat/linregrmeans

FROM UNFITTED TO FITTED MODEL

HOW DOES THE UNFITTED MODEL LOOKS LIKE?

$$\underbrace{\widehat{Price}}_{\hat{y}} = \underbrace{\beta_1}_{m}\underbrace{Sqft}_{x} + \underbrace{\beta_2}_{b}$$

FITTING THE MODEL WITH TIDYMODELS

Note, the data have already been split into `DataTrain` and `DataTest` in the background.

▶ Code

```
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   52509.   64183.      0.818 4.15e- 1
2 Sqft            240.      30.6     7.84  5.67e-12
```

Textbook

UNFITTED MODEL VS FITTED WORKFLOW MODEL

Unfitted Model:

$$\underbrace{\widehat{Price}}_{\hat{y}} = \underbrace{\beta_1}_{m}\underbrace{Sqft}_{x} + \underbrace{\beta_0}_{b}$$

```
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   52509.   64183.      0.818 4.15e- 1
2 Sqft            240.      30.6     7.84  5.67e-12
```

Fitted Model:

$$\underbrace{\widehat{Price}}_{\hat{y}} = \underbrace{240}_{m} \cdot \underbrace{Sqft}_{x} + \underbrace{52509}_{b}$$

Textbook

INTERPRETATION AND SIGNIFICANCE

```
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   52509.    64183.      0.818 4.15e- 1
2 Sqft            240.      30.6     7.84  5.67e-12
```

$$\widehat{Price} = 240 \cdot Sqft + 52509$$
$$(+240) = 240 \cdot (+1) + (+0)$$
$$(+480) = 240 \cdot (+2) + (+0)$$
$$(+720) = 240 \cdot (+3) + (+0)$$

For each extra $Sqft$ the predicted price increases by $240

The variable $Sqft$ is significant. I.e., the probability that the related coefficient β_1 equals zero is extremely small.

HOW DOES THE FITTED MODEL THAT CONSIDERS SQFT IMPROVES THE PREDICTION COMPARED TO A SIMPLE AVERAGE

Look at the simulation again and choose 240 for beta 1:

https://econ.lange-analytics.com/calcat/linregrmeans

EVALUATING PREDICTIVE QUALITY WITH THE TRAINING AND TESTING DATASET

DataTrain:

▶ Code

```
# A tibble: 3 × 3
  .metric .estimator  .estimate
  <chr>   <chr>           <dbl>
1 rmse    standard   230129.
2 rsq     standard        0.385
3 mae     standard   174350.
```

DataTest:

▶ Code

```
# A tibble: 3 × 3
  .metric .estimator  .estimate
  <chr>   <chr>           <dbl>
1 rmse    standard   163476.
2 rsq     standard        0.626
3 mae     standard   132050.
```

UNIVARIATE LINEAR REGRESSION – MOCKUP DATA EXAMPLE

The Regression:

$$\hat{y}_i = \beta_1 x_i + \beta_2$$

$$\widehat{Grade}_i = \beta_1 \cdot StudyTime_i + \beta_2$$

The Goal

Find values for β_1 and β_2 that minimize the prediction errors $(\hat{y}_i - y_i)^2$ or $(\widehat{Grade}_i - Grade_i)^2$

The Data Table

Mockup Training Dataset

	y	x
i	Grade	StudyTime
1	65	2
2	82	3
3	93	7
4	93	8
5	83	4

Textbook

UNIVARIATE LINEAR REGRESSION – MOCKUP DATA EXAMPLE

The Regression:

$$\widehat{Grade}_i = \beta_1 \cdot StudyTime_i + \beta_2$$

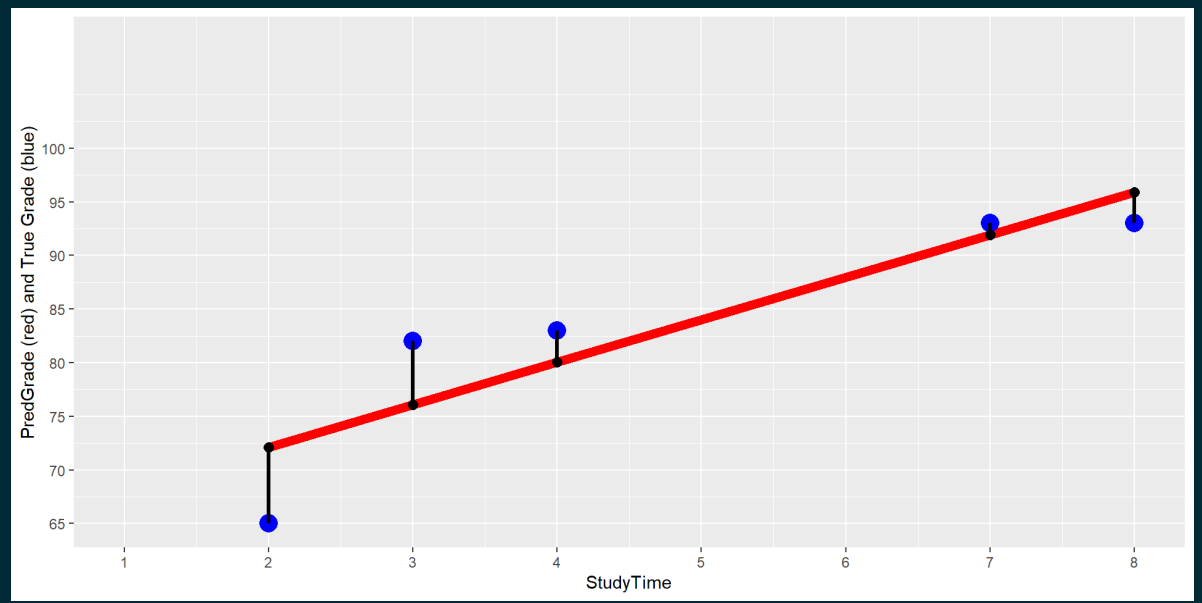
The Goal

Find values for β_0 and β_1 that minimize the prediction errors $(\hat{y}_i - Grade_i)^2$

The Optimal Prediction:

$$\widehat{Grade}_i = 4 \cdot StudyTime_i + 64$$

The Data Diagram



Textbook

HOW TO MEASURE PREDICTION QUALITY

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\Longleftrightarrow$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\overbrace{\beta_1 x_i + \beta_2}^{\text{Prediction } i} - y_i)^2$$

$$\text{Error } i$$

Note, when the data are given (i.e., x_i and y_i are given):
The MSE depends only on the choice of β_1 and β_2

Textbook

HOW TO MEASURE PREDICTION QUALITY WITH THE MSE

$$MSE = \frac{(\beta_1 x_1 + \beta_2 - y_1)^2 + (\beta_1 x_2 + \beta_2 - y_2)^2 + \cdots + (\beta_1 x_5 + \beta_2 - y_5)^2}{5}$$

$$\Longleftrightarrow$$

$$MSE = \frac{1}{5}\Bigg[\underbrace{(\overbrace{\beta_1 \cdot 2 + \beta_2}^{\text{Prediction 1}} - 65)^2}_{\text{Error 1}} + \underbrace{(\overbrace{\beta_1 \cdot 3 + \beta_2}^{\text{Prediction 2}} - 82)^2}_{\text{Error 2}}$$

$$+\underbrace{(\overbrace{\beta_1 \cdot 7 + \beta_2}^{\text{Prediction 3}} - 93)^2}_{\text{Error 3}} + \underbrace{(\overbrace{\beta_1 \cdot 8 + \beta_2}^{\text{Prediction 4}} - 93)^2}_{\text{Error 4}}$$

$$+\underbrace{(\overbrace{\beta_1 \cdot 4 + \beta_2}^{\text{Prediction 5}} - 83)^2}_{\text{Error 6}}\Bigg]$$

Textbook

Textbook

CUSTOM R FUNCTION TO CALCULATE MSE

Function Call:

```r
VecGuessBeta=c(4,64)
ResultMSE=FctMSE(VecGuessBeta, DataMockup)
print(ResultMSE)
```

```
[1] 20.8
```

Function Definition:

▶ Code

HOW TO FIND OPTIMAL VALUES FOR β_1 AND β_2

Method 1:

Calculate optimal values for the parameters (the βs) based on Ordinary Least Squares (OLS) using two formulas. **Note,** this method works only for linear regression.

Method 2:

We can use a **systematic trial and error process**.

METHOD 1: CALCULATE OPTIMAL PARAMETERS (ONLY FOR OLS!)

$$\beta_{1,opt} = \frac{N\sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} = 3.96 \quad \text{and} \quad \beta_{2,opt.} = \frac{\sum_{i=1}^{N} y_i}{N} - \beta_1 \frac{\sum_{i=1}^{N} x}{N}$$

	y	x	y x	x x
i	Grade	StudyTime	GradeXStudyTime	StudyTimeSquared
1	65	2	130	4
2	82	3	246	9
3	93	7	651	49
4	93	8	744	64
5	83	4	332	16

Column Sums

Grade	StudyTime	GradeXStudyTime	StudyTimeSquared
416	24	2103	142

Textbook

METHOD 2: USE A SYSTEMATIC TRIAL AND ERROR PROCESS

- **Manual Trial and Error (aka Brute Force):**
- **Trial and Error with Grid Search (aka Brute Force):**
  1. For a given range of β_1 and β_2 values, build a table with pairs of all combinations of these βs.
  2. Then use our custom `FctMSE()` command to calculate a MSE for each β pair.
  3. Find the β pair with the lowest MSE
- **Optimizer:** Use the R build-in optimizer. Push the start values for β_1 and β_2 together with the data to the optimizer as arguments. The rest is done by the optimizer.
- See the R script in the footnote to see both algorithms in action.

MANUAL TRIAL AND ERROR (AKA BRUTE FORCE)

$$\widehat{Grade}_i = \beta 1 \cdot StudyTime_i + \beta 2$$

R Code | Start Over | Run Code

```
1 GuessBeta1=0
2 GuessBeta2=83
3 MSE=FctMSE(c(GuessBeta1,GuessBeta2), data=DataMockup)
4 cat("The MSE for beta1=",GuessBeta1, "and beta2=",GuessBeta2, "is:", MSE)
```

Textbook

MORE SYSTEMATIC MANUAL TRIAL AND ERROR (STILL BRUTE FORCE)

Average `StudyTime` leads to a prediction of average grade:

$$\overline{StudyTime} = mean(StudyTime) = 4.8$$

$$\overline{Grade} = mean(Grade) = 83.2$$

Therefore:

$$\overline{Grade} = \beta_1 \cdot \overline{StudyTime} + \beta_2$$

$$\beta_2 = \overline{Grade} - \beta_1 \cdot \overline{StudyTime}$$

R Code | Start Over | Run Code

```r
GuessBeta1=0
GuessBeta2=83.2 - GuessBeta1 * 4.8
MSE=FctMSE(c(GuessBeta1,GuessBeta2), data=DataMockup)


cat("The MSE for beta1=",GuessBeta1, "and beta2=",GuessBeta2, "is:", MSE)
```

Textbook

TRIAL AND ERROR WITH GRID SEARCH (AKA BRUTE FORCE)

Guesses for each β:

▶ Code

	Beta1Values	Beta2Values
1	3.0	50
2	3.1	51
3	3.2	52
4	3.3	53
5	3.4	54
6	3.5	55
7	3.6	56
8	3.7	57
9	3.8	58
10	3.9	59
11	4.0	60
12	4.1	61
13	4.2	62
14	4.3	63
15	4.4	64
16	4.5	65
17	4.6	66
18	4.7	67
19	4.8	68
20	4.9	69
21	5.0	70

Grid with all β_1/β_2 combinations with MSE:

▶ Code

	Beta1	Beta2	MSE
1	3.0	50	379.200
2	3.1	50	360.404
3	3.2	50	342.176
4	3.3	50	324.516
5	3.4	50	307.424
6	3.5	50	290.900
7	3.6	50	274.944
8	3.7	50	259.556
9	3.8	50	244.736
10	3.9	50	230.484
11	4.0	50	216.800
12	4.1	50	203.684
13	4.2	50	191.136
14	4.3	50	179.156
15	4.4	50	167.744
16	4.5	50	156.900
17	4.6	50	146.624
18	4.7	50	136.916
19	4.8	50	127.776
20	4.9	50	119.204
21	5.0	50	111.200
22	3.0	51	342.600

Grid with all β_1/β_2 combinations with MSE (sorted):

▶ Code

	Beta1	Beta2	MSE
1	4.0	64	20.800
2	3.8	65	20.936
3	3.9	64	21.044
4	3.9	65	21.084
5	4.2	63	21.096
6	4.1	64	21.124
7	4.1	63	21.164
8	3.7	65	21.356
9	3.7	66	21.476
10	3.6	66	21.504
11	4.3	63	21.596
12	4.3	62	21.716
13	4.0	63	21.800
14	4.0	65	21.800
15	4.4	62	21.824
16	3.8	64	21.856
17	3.8	66	22.016
18	4.2	64	22.016
19	3.5	66	22.100
20	4.2	62	22.176
21	3.5	67	22.300
22	3.6	65	22.344

TRIAL AND ERROR WITH OPTIMIZER (HEURISTIC ALGORITHM, NOT BRUTE FORCE)

R Code | Start Over | Run Code

```r
StartForBeta1Beta2=c(0,83) # 83 is the mean grade

BestBetas=optim(StartForBeta1Beta2, FctMSE, data=DataMockup)

cat("Optimized values for Beta1 and Beta2:", BestBetas$par)
```

MULTIVARIATE OLS WITH A REAL WORLD DATASET

MULTIVARIATE OLS WITH A REAL WORLD DATASET

- King County House Sale dataset (Kaggle 2015). House sales prices from May 2014 to May 2015 for King County in Washington State.
- Several predictor variables.
- We will use all 21,613 observations.

MULTIVARIATE ANALYSIS – THREE PREDICTOR VARIABLES

`Sqft`: Living square footage of the house

`Grade` Indicates the condition of houses (1 (worst) to 13 (best))

`Waterfront`: Is house located at the waterfront (`yes` or `no`)

Unfitted Model:

$$Price_i = \beta_1 Sqft + \beta_2 Grade_i + \beta_3 WaterfrontYes_i + \beta_4$$

MULTIVARIATE REAL WORLD DATASET – SPLITTING

R Code | Start Over | Run Code

```r
DataHousing =
  read.csv("https://ai.lange-analytics.com/data/HousingData.csv")|>
  clean_names("upper_camel") |>
  select(Price, Sqft=SqftLiving, Grade, Waterfront)

set.seed(777)
Split7030=initial_split(0.7,data=DataHousing, strata = Price, breaks = 5)
DataTrain=training(Split7030)
DataTest=testing(Split7030)
cat("DataTrain:")
head(DataTrain)
cat("DataTest:")
head(DataTest)
```

DUMMY AND ONE-HOT ENCODING

One-Hot Encoding

```
# A tibble: 4 × 2
  Waterfront_yes Waterfront_no
           <dbl>         <dbl>
1              0             1
2              0             1
3              1             0
4              0             1
```

One-hot encoding is easier to interpret but causes problems in OLS (dummy trap) because one variable is redundant. We can calculate one variable from the other (*perfect multicollinearity)*:

$$Waterfront_{yes} = 1 - Waterfront_{no}$$

DUMMY AND ONE-HOT ENCODING

Dummy Encoding

We use one variable less than we have categories. Waterfront has two categories. Therefore, we use one variable (e.g., `Waterfront_yes`):

Dummy Encoding Example

```
# A tibble: 4 × 1
  Waterfront_yes
           <dbl>
1              0
2              0
3              1
4              0
```

Note, dummy encoding can be done with `step_dummy()` in a *tidymodels recipe*.

MULTIVARIATE ANALYSIS – BUILDING THE RECIPE

R Code | Start Over | Run Code

```
RecipeHouses=recipe(Price ~ ., data=DataTrain) |>
                    step_dummy(Waterfront)
tidy(RecipeHouses) # prints a tidy form of the recipe
```

Here is how the recipe later on (in the workflow) transforms the data:

```
# A tibble: 6 × 4
   Sqft Grade  Price Waterfront_yes
  <int> <int>  <dbl>          <dbl>
1  1180     7 221900              0
2   770     6 180000              0
3  1200     7 189000              0
4  1250     7 230000              0
5  1070     7 252700              0
6  1220     7 240000              0
```

MULTIVARIATE ANALYSIS – BUILDING THE MODEL DESIGN

Unfitted Model:

R Code | Start Over | Run Code

```r
ModelDesignHouses=linear_reg() |>
  set_engine("lm") |>
  set_mode("regression")
print(ModelDesignHouses)
```

MULTIVARIATE ANALYSIS – CREATING WORKFLOW & FITTING TO THE TRAINING DATA

R Code | Start Over | Run Code

```
WFModelHouses = workflow() |>
      add_recipe(RecipeHouses) |>
      add_model(ModelDesignHouses) |>
      fit(DataTrain)
print(WFModelHouses)
```

MULTIVARIATE ANALYSIS – PREDICTING TESTING DATA AND METRICS

R Code | Start Over | Run Code

```r
DataTestWithPredictions = augment(WFModelHouses, new_data=DataTest)
metrics(DataTestWithPredictions, truth=Price, estimate=.pred)
```