**Step Five: Modeling**

**Modeling** is the fifth step in the Data Science Method. During this step, you'll leverage your cleaned and processed data to make predictive insights.

Please note that modeling involves both model training and selection, as well as model deployment. This subunit will provide a high-level overview of model training and selection but will not teach model deployment (you'll learn all about this topic in a later unit).



[Write](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Fmedium.com%2Fnew-story&source=---two_column_layout_nav-----------------------new_post_sidenav-----------)

[Sign up](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fmodelling-e096495d8d70&source=post_page---two_column_layout_nav-----------------------global_nav-----------)

[Sign In](https://medium.com/m/signin?operation=login&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fmodelling-e096495d8d70&source=post_page---two_column_layout_nav-----------------------global_nav-----------)

[[Towards Data Science](https://towardsdatascience.com/?source=post_page-----e096495d8d70--------------------------------)](https://towardsdatascience.com/?source=post_page-----e096495d8d70--------------------------------)

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----e096495d8d70--------------------------------)

[[](https://guymaskall.medium.com/?source=post_page-----e096495d8d70--------------------------------)](https://guymaskall.medium.com/?source=post_page-----e096495d8d70--------------------------------)

[Guy Maskall](https://guymaskall.medium.com/?source=post_page-----e096495d8d70--------------------------------)

Jul 15, 2020

·

2 min read

·

# Modelling

## Often simplistically reduced to merely training a model, this is really about connecting predictions to the real world.



Image by [Gerd Altmann](https://pixabay.com/users/geralt-9301/?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=3706562) from [Pixabay](https://pixabay.com/?utm_source=link-attribution&utm_medium=referral&utm_campaign=image&utm_content=3706562)

# Finalising the model

We’ve discussed using cross-validation to help us with the selection process of picking the best algorithm for our problem. It is worth noting that with, say, five-fold cross-validation, we train and test a model five times. Each time the model is trained on 80% of the data and tested on the remaining 20%. But which of these trained models do we end up with? The answer is actually that the final model is retrained on the full training data set. After all, we’ve done our holding back of data to assess model performance; we don’t have to keep a partition held back any longer. Why waste it? We can use that extra 20% to make a model that should be that much more robust because it’s been trained on more data.

Similarly, once we’ve done all of our cross-validation and model selection, and checked the model performance using our ultimate hold-out test set (that the entire process has not seen before), we can actually fold even that set of data into our training to hopefully yield a slightly better result. We just need to remember that we can’t make any further adjustments to our algorithm design. This is an important point to note. We cannot decide to make our model just a bit more complex to just do a bit better on this particular sample. We cannot decide that a little bit more, or less, regularization is the right thing to do. We might lament that we have lost our independent assessment of our algorithm’s performance, but if we were happy to accept our test set was good data and so gave us a valid check on performance, we should be happy to incorporate that data into refining our model’s understanding of the world.

# Inference

The preceding step, where we perform pre-processing and training, has actually done what many texts treat as modelling. For us here, modelling is less the development of a model, and more the application of a trained model to perform inference for modelling the behaviour of a system. After all, the fundamental reason we build predictive machine learning models is typically to model what would happen under a particular set of circumstances which have either not yet occurred or have occurred but for which we do not know the actual result.

# About this article

This is the sixth article of a linked series written to provide a straightforward introduction to getting started with the data science process. You can find the introduction [here](https://medium.com/@guymaskall/the-data-science-method-dsm-35200eb4984), the previous article [here](https://medium.com/@guymaskall/pre-processing-and-training-data-d16cc12dfbac), and the next article in the series [here](https://medium.com/@guymaskall/documentation-511c1fcb8fe)

# 5 Modeling

## 5.1 Contents

* [5 Modeling](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5_Modeling)
  + [5.1 Contents](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.1_Contents)
  + [5.2 Introduction](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.2_Introduction)
  + [5.3 Imports](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.3_Imports)
  + [5.4 Load Model](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.4_Load_Model)
  + [5.5 Load Data](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.5_Load_Data)
  + [5.6 Refit Model On All Available Data (excluding Big Mountain)](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.6_Refit_Model_On_All_Available_Data_(excluding_Big_Mountain))
  + [5.7 Calculate Expected Big Mountain Ticket Price From The Model](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.7_Calculate_Expected_Big_Mountain_Ticket_Price_From_The_Model)
  + [5.8 Big Mountain Resort In Market Context](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8_Big_Mountain_Resort_In_Market_Context)
    - [5.8.1 Ticket price](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.1_Ticket_price)
    - [5.8.2 Vertical drop](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.2_Vertical_drop)
    - [5.8.3 Snow making area](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.3_Snow_making_area)
    - [5.8.4 Total number of chairs](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.4_Total_number_of_chairs)
    - [5.8.5 Fast quads](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.5_Fast_quads)
    - [5.8.6 Runs](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.6_Runs)
    - [5.8.7 Longest run](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.7_Longest_run)
    - [5.8.8 Trams](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.8_Trams)
    - [5.8.9 Skiable terrain area](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.8.9_Skiable_terrain_area)
  + [5.9 Modeling scenarios](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.9_Modeling_scenarios)
    - [5.9.1 Scenario 1](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.9.1_Scenario_1)
    - [5.9.2 Scenario 2](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.9.2_Scenario_2)
    - [5.9.3 Scenario 3](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.9.3_Scenario_3)
    - [5.9.4 Scenario 4](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.9.4_Scenario_4)
  + [5.10 Summary](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.10_Summary)
  + [5.11 Further work](https://github.com/charriemascio/Springboard2/blob/2237553a0f3c0b2d6b24f50104b7c502d8f4f493/Notebooks/#5.11_Further_work)

## 5.2 Introduction

In this notebook, we now take our model for ski resort ticket price and leverage it to gain some insights into what price Big Mountain's facilities might actually support as well as explore the sensitivity of changes to various resort parameters. Note that this relies on the implicit assumption that all other resorts are largely setting prices based on how much people value certain facilities. Essentially this assumes prices are set by a free market.

We can now use our model to gain insight into what Big Mountain's ideal ticket price could/should be, and how that might change under various scenarios.

## 5.3 Imports

import pandas as pd

import numpy as np

import os

import pickle

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import \_\_version\_\_ as sklearn\_version

from sklearn.model\_selection import cross\_validate

## 5.4 Load Model

# This isn't exactly production-grade, but a quick check for development

# These checks can save some head-scratching in development when moving from

# one python environment to another, for example

expected\_model\_version = '1.0'

model\_path = '../models/ski\_resort\_pricing\_model.pkl'

if os.path.exists(model\_path):

with open(model\_path, 'rb') as f:

model = pickle.load(f)

if model.version != expected\_model\_version:

print("Expected model version doesn't match version loaded")

if model.sklearn\_version != sklearn\_version:

print("Warning: model created under different sklearn version")

else:

print("Expected model not found")

## 5.5 Load Data

ski\_data = pd.read\_csv('../data/ski\_data\_step3\_features.csv')

big\_mountain = ski\_data[ski\_data.Name == 'Big Mountain Resort']

big\_mountain.T

|  | **124** |
| --- | --- |
| **Name** | Big Mountain Resort |
| **Region** | Montana |
| **state** | Montana |
| **summit\_elev** | 6817 |
| **vertical\_drop** | 2353 |
| **base\_elev** | 4464 |
| **trams** | 0 |
| **fastSixes** | 0 |
| **fastQuads** | 3 |
| **quad** | 2 |
| **triple** | 6 |
| **double** | 0 |
| **surface** | 3 |
| **total\_chairs** | 14 |
| **Runs** | 105 |
| **TerrainParks** | 4 |
| **LongestRun\_mi** | 3.3 |
| **SkiableTerrain\_ac** | 3000 |
| **Snow Making\_ac** | 600 |
| **daysOpenLastYear** | 123 |
| **yearsOpen** | 72 |
| **averageSnowfall** | 333 |
| **AdultWeekend** | 81 |
| **projectedDaysOpen** | 123 |
| **NightSkiing\_ac** | 600 |
| **resorts\_per\_state** | 12 |
| **resorts\_per\_100kcapita** | 1.12278 |
| **resorts\_per\_100ksq\_mile** | 8.16104 |
| **resort\_skiable\_area\_ac\_state\_ratio** | 0.140121 |
| **resort\_days\_open\_state\_ratio** | 0.129338 |
| **resort\_terrain\_park\_state\_ratio** | 0.148148 |
| **resort\_night\_skiing\_state\_ratio** | 0.84507 |
| **total\_chairs\_runs\_ratio** | 0.133333 |
| **total\_chairs\_skiable\_ratio** | 0.00466667 |
| **fastQuads\_runs\_ratio** | 0.0285714 |
| **fastQuads\_skiable\_ratio** | 0.001 |

## 5.6 Refit Model On All Available Data (excluding Big Mountain)

This next step requires some careful thought. We want to refit the model using all available data. But should we include Big Mountain data? On the one hand, we are not trying to estimate model performance on a previously unseen data sample, so theoretically including Big Mountain data should be fine. One might first think that including Big Mountain in the model training would, if anything, improve model performance in predicting Big Mountain's ticket price. But here's where our business context comes in. The motivation for this entire project is based on the sense that Big Mountain needs to adjust its pricing. One way to phrase this problem: we want to train a model to predict Big Mountain's ticket price based on data from all the other resorts! We don't want Big Mountain's current price to bias this. We want to calculate a price based only on its competitors.

X = ski\_data.loc[ski\_data.Name != "Big Mountain Resort", model.X\_columns]

y = ski\_data.loc[ski\_data.Name != "Big Mountain Resort", 'AdultWeekend']

len(X), len(y)

(276, 276)

model.fit(X, y)

Pipeline(steps=[('simpleimputer', SimpleImputer(strategy='median')),

('standardscaler', None),

('randomforestregressor',

RandomForestRegressor(n\_estimators=69, random\_state=47))])

cv\_results = cross\_validate(model, X, y, scoring='neg\_mean\_absolute\_error', cv=5, n\_jobs=-1)

cv\_results['test\_score']

array([-12.09690217, -9.30247694, -11.41595784, -8.10096706,

-11.04942819])

mae\_mean, mae\_std = np.mean(-1 \* cv\_results['test\_score']), np.std(-1 \* cv\_results['test\_score'])

mae\_mean, mae\_std

(10.393146442687748, 1.4712769116280346)

These numbers will inevitably be different to those in the previous step that used a different training data set. They should, however, be consistent. It's important to appreciate that estimates of model performance are subject to the noise and uncertainty of data!

## 5.7 Calculate Expected Big Mountain Ticket Price From The Model

X\_bm = ski\_data.loc[ski\_data.Name == "Big Mountain Resort", model.X\_columns]

y\_bm = ski\_data.loc[ski\_data.Name == "Big Mountain Resort", 'AdultWeekend']

bm\_pred = model.predict(X\_bm).item()

y\_bm = y\_bm.values.item()

print(f'Big Mountain Resort modelled price is ${bm\_pred:.2f}, actual price is ${y\_bm:.2f}.')

print(f'Even with the expected mean absolute error of ${mae\_mean:.2f}, this suggests there is room for an increase.')

Big Mountain Resort modelled price is $95.87, actual price is $81.00.

Even with the expected mean absolute error of $10.39, this suggests there is room for an increase.

This result should be looked at optimistically and doubtfully! The validity of our model lies in the assumption that other resorts accurately set their prices according to what the market (the ticket-buying public) supports. The fact that our resort seems to be charging that much less that what's predicted suggests our resort might be undercharging. But if ours is mispricing itself, are others? It's reasonable to expect that some resorts will be "overpriced" and some "underpriced." Or if resorts are pretty good at pricing strategies, it could be that our model is simply lacking some key data? Certainly we know nothing about operating costs, for example, and they would surely help.

## 5.8 Big Mountain Resort In Market Context

Features that came up as important in the modeling (not just our final, random forest model) included:

* vertical\_drop
* Snow Making\_ac
* total\_chairs
* fastQuads
* Runs
* LongestRun\_mi
* trams
* SkiableTerrain\_ac

A handy glossary of skiing terms can be found on the [ski.com](https://www.ski.com/ski-glossary) site. Some potentially relevant contextual information is that vertical drop, although nominally the height difference from the summit to the base, is generally taken from the highest [lift-served](http://verticalfeet.com/) point.

It's often useful to define custom functions for visualizing data in meaningful ways. The function below takes a feature name as an input and plots a histogram of the values of that feature. It then marks where Big Mountain sits in the distribution by marking Big Mountain's value with a vertical line using matplotlib's [axvline](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axvline.html) function. It also performs a little cleaning up of missing values and adds descriptive labels and a title.

#Code task 1#

#Add code to the `plot\_compare` function that displays a vertical, dashed line

#on the histogram to indicate Big Mountain's position in the distribution

#Hint: plt.axvline() plots a vertical line, its position for 'feature1'

#would be `big\_mountain['feature1'].values, we'd like a red line, which can be

#specified with c='r', a dashed linestyle is produced by ls='--',

#and it's nice to give it a slightly reduced alpha value, such as 0.8.

#Don't forget to give it a useful label (e.g. 'Big Mountain') so it's listed

#in the legend.

def plot\_compare(feat\_name, description, state=None, figsize=(10, 5)):

"""Graphically compare distributions of features.

Plot histogram of values for all resorts and reference line to mark

Big Mountain's position.

Arguments:

feat\_name - the feature column name in the data

description - text description of the feature

state - select a specific state (None for all states)

figsize - (optional) figure size

"""

plt.subplots(figsize=figsize)

# quirk that hist sometimes objects to NaNs, sometimes doesn't

# filtering only for finite values tidies this up

if state is None:

ski\_x = ski\_data[feat\_name]

else:

ski\_x = ski\_data.loc[ski\_data.state == state, feat\_name]

ski\_x = ski\_x[np.isfinite(ski\_x)]

plt.hist(ski\_x, bins=30)

plt.\_\_\_(x=big\_mountain[feat\_name].\_\_\_, c=\_\_\_, ls=\_\_\_, alpha=0.8, label=\_\_\_)

plt.xlabel(description)

plt.ylabel('frequency')

plt.title(description + ' distribution for resorts in market share')

plt.legend()

### 5.8.1 Ticket price

Look at where Big Mountain sits overall amongst all resorts for price and for just other resorts in Montana.

plot\_compare('AdultWeekend', 'Adult weekend ticket price ($)')

Chart, histogram

Description automatically generated

plot\_compare('AdultWeekend', 'Adult weekend ticket price ($) - Montana only', state='Montana')

A picture containing bar chart

Description automatically generated

### 5.8.2 Vertical drop

plot\_compare('vertical\_drop', 'Vertical drop (feet)')

Chart, histogram

Description automatically generated

Big Mountain is doing well for vertical drop, but there are still quite a few resorts with a greater drop.

### 5.8.3 Snow making area

plot\_compare('Snow Making\_ac', 'Area covered by snow makers (acres)')

Chart, histogram

Description automatically generated

Big Mountain is very high up the league table of snow making area.

### 5.8.4 Total number of chairs

plot\_compare('total\_chairs', 'Total number of chairs')

Chart, histogram

Description automatically generated

Big Mountain has amongst the highest number of total chairs, resorts with more appear to be outliers.

### 5.8.5 Fast quads

plot\_compare('fastQuads', 'Number of fast quads')

A picture containing histogram

Description automatically generated

Most resorts have no fast quads. Big Mountain has 3, which puts it high up that league table. There are some values much higher, but they are rare.

### 5.8.6 Runs

plot\_compare('Runs', 'Total number of runs')

Chart, histogram

Description automatically generated

Big Mountain compares well for the number of runs. There are some resorts with more, but not many.

### 5.8.7 Longest run

plot\_compare('LongestRun\_mi', 'Longest run length (miles)')

Chart, histogram

Description automatically generated

Big Mountain has one of the longest runs. Although it is just over half the length of the longest, the longer ones are rare.

### 5.8.8 Trams

plot\_compare('trams', 'Number of trams')

Chart

Description automatically generated

The vast majority of resorts, such as Big Mountain, have no trams.

### 5.8.9 Skiable terrain area

plot\_compare('SkiableTerrain\_ac', 'Skiable terrain area (acres)')

A picture containing shape

Description automatically generated

Big Mountain is amongst the resorts with the largest amount of skiable terrain.

## 5.9 Modeling scenarios

Big Mountain Resort has been reviewing potential scenarios for either cutting costs or increasing revenue (from ticket prices). Ticket price is not determined by any set of parameters; the resort is free to set whatever price it likes. However, the resort operates within a market where people pay more for certain facilities, and less for others. Being able to sense how facilities support a given ticket price is valuable business intelligence. This is where the utility of our model comes in.

The business has shortlisted some options:

1. Permanently closing down up to 10 of the least used runs. This doesn't impact any other resort statistics.
2. Increase the vertical drop by adding a run to a point 150 feet lower down but requiring the installation of an additional chair lift to bring skiers back up, without additional snow making coverage
3. Same as number 2, but adding 2 acres of snow making cover
4. Increase the longest run by 0.2 mile to boast 3.5 miles length, requiring an additional snow making coverage of 4 acres

The expected number of visitors over the season is 350,000 and, on average, visitors ski for five days. Assume the provided data includes the additional lift that Big Mountain recently installed.

expected\_visitors = 350\_000

all\_feats = ['vertical\_drop', 'Snow Making\_ac', 'total\_chairs', 'fastQuads',

'Runs', 'LongestRun\_mi', 'trams', 'SkiableTerrain\_ac']

big\_mountain[all\_feats]

|  | **vertical\_drop** | **Snow Making\_ac** | **total\_chairs** | **fastQuads** | **Runs** | **LongestRun\_mi** | **trams** | **SkiableTerrain\_ac** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **124** | 2353 | 600.0 | 14 | 3 | 105.0 | 3.3 | 0 | 3000.0 |

#Code task 2#

#In this function, copy the Big Mountain data into a new data frame

#(Note we use .copy()!)

#And then for each feature, and each of its deltas (changes from the original),

#create the modified scenario dataframe (bm2) and make a ticket price prediction

#for it. The difference between the scenario's prediction and the current

#prediction is then calculated and returned.

#Complete the code to increment each feature by the associated delta

def predict\_increase(features, deltas):

"""Increase in modelled ticket price by applying delta to feature.

Arguments:

features - list, names of the features in the ski\_data dataframe to change

deltas - list, the amounts by which to increase the values of the features

Outputs:

Amount of increase in the predicted ticket price

"""

bm2 = X\_bm.copy()

for f, d in zip(features, deltas):

bm2[\_\_\_] += \_\_\_

return model.predict(bm2).item() - model.predict(X\_bm).item()

### 5.9.1 Scenario 1

Close up to 10 of the least used runs. The number of runs is the only parameter varying.

[i for i in range(-1, -11, -1)]

[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]

runs\_delta = [i for i in range(-1, -11, -1)]

price\_deltas = [predict\_increase(['Runs'], [delta]) for delta in runs\_delta]

price\_deltas

[0.0,

-0.4057971014492807,

-0.6666666666666714,

-0.6666666666666714,

-0.6666666666666714,

-1.2608695652173907,

-1.2608695652173907,

-1.2608695652173907,

-1.7101449275362341,

-1.8115942028985472]

#Code task 3#

#Create two plots, side by side, for the predicted ticket price change (delta) for each

#condition (number of runs closed) in the scenario and the associated predicted revenue

#change on the assumption that each of the expected visitors buys 5 tickets

#There are two things to do here:

#1 - use a list comprehension to create a list of the number of runs closed from `runs\_delta`

#2 - use a list comprehension to create a list of predicted revenue changes from `price\_deltas`

runs\_closed = [-1 \* \_\_\_ for \_\_\_ in runs\_delta] #1

fig, ax = plt.subplots(1, 2, figsize=(10, 5))

fig.subplots\_adjust(wspace=0.5)

ax[0].plot(runs\_closed, price\_deltas, 'o-')

ax[0].set(xlabel='Runs closed', ylabel='Change ($)', title='Ticket price')

revenue\_deltas = [5 \* expected\_visitors \* \_\_\_ for \_\_\_ in \_\_\_] #2

ax[1].plot(runs\_closed, revenue\_deltas, 'o-')

ax[1].set(xlabel='Runs closed', ylabel='Change ($)', title='Revenue');

The model says closing one run makes no difference. Closing 2 and 3 successively reduces support for ticket price and so revenue. If Big Mountain closes down 3 runs, it seems they may as well close down 4 or 5 as there's no further loss in ticket price. Increasing the closures down to 6 or more leads to a large drop.

### 5.9.2 Scenario 2

In this scenario, Big Mountain is adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift.

#Code task 4#

#Call `predict\_increase` with a list of the features 'Runs', 'vertical\_drop', and 'total\_chairs'

#and associated deltas of 1, 150, and 1

ticket2\_increase = \_\_\_(['Runs', \_\_\_, \_\_\_], [1, \_\_\_, \_\_\_])

revenue2\_increase = 5 \* expected\_visitors \* ticket2\_increase

print(f'This scenario increases support for ticket price by ${ticket2\_increase:.2f}')

print(f'Over the season, this could be expected to amount to ${revenue2\_increase:.0f}')

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

### 5.9.3 Scenario 3

In this scenario, you are repeating the previous one but adding 2 acres of snow making.

#Code task 5#

#Repeat scenario 2 conditions, but add an increase of 2 to `Snow Making\_ac`

ticket3\_increase = predict\_increase(['Runs', 'vertical\_drop', 'total\_chairs', \_\_\_], [1, 150, 1, \_\_\_])

revenue3\_increase = 5 \* expected\_visitors \* ticket3\_increase

print(f'This scenario increases support for ticket price by ${ticket3\_increase:.2f}')

print(f'Over the season, this could be expected to amount to ${revenue3\_increase:.0f}')

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

Such a small increase in the snow making area makes no difference!

### 5.9.4 Scenario 4

This scenario calls for increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability.

#Code task 6#

#Predict the increase from adding 0.2 miles to `LongestRun\_mi` and 4 to `Snow Making\_ac`

predict\_increase([\_\_\_, \_\_\_], [\_\_\_, \_\_\_])

No difference whatsoever. Although the longest run feature was used in the linear model, the random forest model (the one we chose because of its better performance) only has longest run way down in the feature importance list.

## 5.10 Summary

**Q: 1** Write a summary of the results of modeling these scenarios. Start by starting the current position; how much does Big Mountain currently charge? What does your modelling suggest for a ticket price that could be supported in the marketplace by Big Mountain's facilities? How would you approach suggesting such a change to the business leadership? Discuss the additional operating cost of the new chair lift per ticket (on the basis of each visitor on average buying 5 day tickets) in the context of raising prices to cover this. For future improvements, state which, if any, of the modeled scenarios you'd recommend for further consideration. Suggest how the business might test, and progress, with any run closures.

**A: 1** Your answer here

## 5.11 Further work

**Q: 2** What next? Highlight any deficiencies in the data that hampered or limited this work. The only price data in our dataset were ticket prices. You were provided with information about the additional operating cost of the new chair lift, but what other cost information would be useful? Big Mountain was already fairly high on some of the league charts of facilities offered, but why was its modeled price so much higher than its current price? Would this mismatch come as a surprise to the business executives? How would you find out? Assuming the business leaders felt this model was useful, how would the business make use of it? Would you expect them to come to you every time they wanted to test a new combination of parameters in a scenario? We hope you would have better things to do, so how might this model be made available for business analysts to use and explore?

## 5.2 Introduction

In this notebook, we now take our model for ski resort ticket price and leverage it to gain some insights into what price Big Mountain's facilities might actually support as well as explore the sensitivity of changes to various resort parameters. Note that this relies on the implicit assumption that all other resorts are largely setting prices based on how much people value certain facilities. Essentially this assumes prices are set by a free market.

We can now use our model to gain insight into what Big Mountain's ideal ticket price could/should be, and how that might change under various scenarios.

## 5.10 Summary

**Q: 1** Write a summary of the results of modeling these scenarios. Start by starting the current position; how much does Big Mountain currently charge? What does your modelling suggest for a ticket price that could be supported in the marketplace by Big Mountain's facilities? How would you approach suggesting such a change to the business leadership? Discuss the additional operating cost of the new chair lift per ticket (on the basis of each visitor on average buying 5 day tickets) in the context of raising prices to cover this. For future improvements, state which, if any, of the modeled scenarios you'd recommend for further consideration. Suggest how the business might test, and progress, with any run closures.

**A: 1** Your answer here

## 5.11 Further work

**Q: 2** What next? Highlight any deficiencies in the data that hampered or limited this work. The only price data in our dataset were ticket prices. You were provided with information about the additional operating cost of the new chair lift, but what other cost information would be useful? Big Mountain was already fairly high on some of the league charts of facilities offered, but why was its modeled price so much higher than its current price? Would this mismatch come as a surprise to the business executives? How would you find out? Assuming the business leaders felt this model was useful, how would the business make use of it? Would you expect them to come to you every time they wanted to test a new combination of parameters in a scenario? We hope you would have better things to do, so how might this model be made available for business analysts to use and explore?

**5.10**

The Big Mountain currently charges adult weekend price for $81.

This project is based on Big Mountain wanting to adjust its pricing. The problem is addressed by training a model to predict Big Mountain's ticket price based on data from all the other resorts. We don't want Big Mountain's current price to bias this, a price was calculated based only on its competitors.

Model was fitted using the pipeline, steps that imputes the median, random forest regressor, cross validate and results were MAE mean of 10.393, and MAE std of 1.471

These numbers will inevitably be different to those in the previous step that used a different training data set. They should, however, be consistent. It's important to appreciate that estimates of model performance are subject to the noise and uncertainty of data.

Next, calculating expected Big Mountain ticket price from the model and this was performed using the model predict method. Big Mountain Resort modelled price is $95.87, actual price is $81.00. Even with the expected mean absolute error of $10.39, this suggests there is room for an increase.

This result should be looked at optimistically and cautiously. The validity of our model lies in the assumption that other resorts accurately set their prices according to what the market (the ticket-buying public) supports. The fact that our resort seems to be charging that much less than what's predicted suggests our resort might be undercharging. The question is, are the other resorts also mispricing or do they have good pricing strategies? It's reasonable to expect that some resorts will be "overpriced" and some "underpriced." It could be that our model is simply lacking some key data. Knowing about operating costs might help.

From the modelling data, features that came up as important not just our final, random forest model, include vertical drop, snow making in acres, total chairs, fast quads, runs, longest run in miles and trams.

* In the US market, it looks like Montana (Big Mountain) is in the upper half in resort prices. In Montana (Big Mountain) is the most expensive resort price at $81.
* Big Mountain is doing well for vertical drop(2353 ft), but there are still quite a few resorts with a greater drop.
* Big Mountain is very high up the league table of snow making area (600 acres) among distribution for resorts in market share.
* Big Mountain has amongst the highest number of total chairs (14), resorts with more appear to be outliers.
* Most resorts have no fast quads. Big Mountain has 3, which puts it high up that league table. There are some values much higher, but they are rare.
* Big Mountain compares well for the number of runs (105). There are some resorts with more, but not many.
* Big Mountain has one of the longest runs (3.3 miles). Although it is just over half the length of the longest, the longer ones are rare.
* The vast majority of resorts, such as Big Mountain, have no trams.
* Big Mountain is amongst the resorts with the largest amount of skiable terrain (3000 acres).

The modelling suggests a ticket price $95.87 from actual price $81, there is room for increase that could be supported in the marketplace by Big Mountain based on the positive features/facilities that Big Mountain has to offer.

Big Mountain Resort has been reviewing potential scenarios for either cutting costs or increasing revenue (from ticket prices). Ticket price is not determined by any set of parameters; the resort is free to set whatever price it likes. However, the resort operates within a market where people pay more for certain facilities, and less for others. Being able to sense how facilities support a given ticket price is valuable business intelligence. This is where the utility of our model comes in.

The business has shortlisted some options:

1. Permanently closing up to 10 of the least used runs. This doesn't impact any other resort statistics.
2. Increase the vertical drop by adding a run to a point 150 feet lower down but requiring the installation of an additional chair lift to bring skiers back up, without additional snow making coverage. Making vertical drop 2503 ft.
3. Same as number 2 (vertical drop to 2503 ft), but adding 2 acres of snow making cover. Making it 602 acres.
4. Increase the longest run by 0.2 mile to boast 3.5 miles length, requiring an additional snow making coverage of 4 acres

**Testing**

The following prediction tests were done, taking into account the following:

The expected number of visitors over the season is 350,000 and, on average, visitors ski for five days. Assume the provided data includes the additional lift that Big Mountain recently installed.

Increase in modelled ticket price applying the amounts by which to increase the values of the features. This will output the amount in increase in the predicted price.

**Scenario 1**

Close 1 of the least used runs. The number of runs is the only parameter varying.

Price deltas results: Negative result for all features means people would want to decrease the price when closing more than one run.

The model says closing one run makes no difference in the price. Closing 2 or more reduces support for ticket price and revenue. The model shows a couple of plateaus in the closures. Closing 3 has the same effect as closing 4 or 5. But the operational cost savings may offset the loss in ticket prices of $0.67. We also see a large drop to the next plateau of 6, 7, or 8 closures of $1.26 in price reduction.

**Scenario 2**

In this scenario, Big Mountain is adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift.

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

The question to be investigated is, does the added revenue cover the expenses of extending a run by the 150ft drop, including possible new lift to the new starting point. Also, is there a run that is extendable with property and geographic limitations within the overall property?

**Scenario 3**

In this scenario, you are repeating the previous one but adding 2 acres of snow making.

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

Such a small increase in the snow making area makes no difference.

**Scenario 4**

This scenario calls for increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability.

Predict increase was 0. There is no effect on ticket price or revenue.

No difference whatsoever. Although the longest run feature was used in the linear model, the random forest model (the one we chose because of its better performance) only has longest runway down in the feature importance list.

**Conclusion**

Based on the modeled scenarios recommended for further consideration, can be summarized as follows:

Overall, closing 1 run will not dramatically affect ticket price or revenue. Adding a run, increasing the vertical drop by 150 ft and an additional chair lift will support increasing the ticket price by $1.99 and having a predicted revenue of $3474638 over a season. Adding a run, increasing the vertical drop by 150 ft, possibly adding a chair lift, and adding 2 acres of snow making area resulted in supporting increase in ticket price($1.99) and predicted revenues per season of $3474638 but the addition of the 2 acres snow making area does not make a difference. Lastly, adding 0.2 miles to the longest run and guaranteeing snow coverage by adding 4 acres of snow making area did not make a difference.

I would recommend a combination of scenario 1 and 2. Scenario 1 tells us that by closing one run will not affect the ticket price or revenue. But losing one run may also have operational cost savings that can be applied to the bottom line. Modifying a run when considering scenario 2 to adding a vertical drop of 150 ft. This indicates support for an increase in ticket prices and revenues. This scenario will also entail additional investigation into the cost of adding more vertical drop, possible redirection of a run and answer the question of the feasibility of physically adding the drop needs to be investigated.

Maybe further testing additional run closures against the other features in the training and test data plus the predicted number of visitors will show us more information on which additional features will support the ticket price and revenue increase.

**5.11**

We only have data calculations of ticket prices based only on the competitors of Big Mountain resort with the assumption that prices are set according to what the market supports. According to the model, our resort might be charging less than predicted. The question is, is the model we are using lacking any key data, like operating costs of the resorts? This might be something to be investigated further, which may have limited this model.

Aside from the operating cost of the additional chair lift as mentioned, another useful piece of data would be operating cost for the other facilities/features that this resort has to offer. This would include overall operating cost, source of general revenues of the resort and which facilities were doing well or not.

Based on the data, the resort is basing their ticket price according to what the market will support in their area.

Asking the main stakeholders as to what they based their original resort ticket prices might be a way of finding out the reasons why the business executives chose the ticket pricing that they have.

If this model is useful to the business, then they could apply the increase in price as recommended by the data. Any new parameters that are related to the present business problem may be tested but if parameters that are not directly related may be reviewed if it makes sense to add it or make a new project to investigate further.

It would be better to make recommendations based on parameters that the business leaders may have used in the past, to base any future business decisions.

This information could be explored and used by business analysts through Kaggle with using a different name if this is allowed by the business.

Python 3 (ipykernel)

* [File](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Edit](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [View](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Insert](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Cell](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Kernel](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Widgets](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)
* [Help](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb)

# 5 Modeling

## 5.1 Contents

* [5 Modeling](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5_Modeling)
  + [5.1 Contents](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.1_Contents)
  + [5.2 Introduction](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.2_Introduction)
  + [5.3 Imports](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.3_Imports)
  + [5.4 Load Model](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.4_Load_Model)
  + [5.5 Load Data](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.5_Load_Data)
  + [5.6 Refit Model On All Available Data (excluding Big Mountain)](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.6_Refit_Model_On_All_Available_Data_(excluding_Big_Mountain))
  + [5.7 Calculate Expected Big Mountain Ticket Price From The Model](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.7_Calculate_Expected_Big_Mountain_Ticket_Price_From_The_Model)
  + [5.8 Big Mountain Resort In Market Context](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8_Big_Mountain_Resort_In_Market_Context)
    - [5.8.1 Ticket price](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.1_Ticket_price)
    - [5.8.2 Vertical drop](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.2_Vertical_drop)
    - [5.8.3 Snow making area](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.3_Snow_making_area)
    - [5.8.4 Total number of chairs](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.4_Total_number_of_chairs)
    - [5.8.5 Fast quads](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.5_Fast_quads)
    - [5.8.6 Runs](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.6_Runs)
    - [5.8.7 Longest run](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.7_Longest_run)
    - [5.8.8 Trams](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.8_Trams)
    - [5.8.9 Skiable terrain area](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.8.9_Skiable_terrain_area)
  + [5.9 Modeling scenarios](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.9_Modeling_scenarios)
    - [5.9.1 Scenario 1](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.9.1_Scenario_1)
    - [5.9.2 Scenario 2](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.9.2_Scenario_2)
    - [5.9.3 Scenario 3](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.9.3_Scenario_3)
    - [5.9.4 Scenario 4](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.9.4_Scenario_4)
  + [5.10 Summary](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.10_Summary)
  + [5.11 Further work](http://localhost:8890/notebooks/Desktop/Springboard/GitHub/Springboard2/Notebooks/05_modeling.ipynb#5.11_Further_work)

## 5.2 Introduction

In this notebook, we now take our model for ski resort ticket price and leverage it to gain some insights into what price Big Mountain's facilities might actually support as well as explore the sensitivity of changes to various resort parameters. Note that this relies on the implicit assumption that all other resorts are largely setting prices based on how much people value certain facilities. Essentially this assumes prices are set by a free market.

We can now use our model to gain insight into what Big Mountain's ideal ticket price could/should be, and how that might change under various scenarios.

## 5.3 Imports



import pandas as pd

import numpy as np

import os

import pickle

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import \_\_version\_\_ as sklearn\_version

from sklearn.model\_selection import cross\_validate

## 5.4 Load Model



# This isn't exactly production-grade, but a quick check for development

# These checks can save some head-scratching in development when moving from

# one python environment to another, for example

expected\_model\_version = '1.0'

model\_path = '../models/ski\_resort\_pricing\_model.pkl'

if os.path.exists(model\_path):

with open(model\_path, 'rb') as f:

model = pickle.load(f)

if model.version != expected\_model\_version:

print("Expected model version doesn't match version loaded")

if model.sklearn\_version != sklearn\_version:

print("Warning: model created under different sklearn version")

else:

print("Expected model not found")

Warning: model created under different sklearn version

## 5.5 Load Data



ski\_data = pd.read\_csv('../data/ski\_data\_step3\_features.csv')



big\_mountain = ski\_data[ski\_data.Name == 'Big Mountain Resort']



big\_mountain.T

|  | **124** |
| --- | --- |
| **Name** | Big Mountain Resort |
| **Region** | Montana |
| **state** | Montana |
| **summit\_elev** | 6817 |
| **vertical\_drop** | 2353 |
| **base\_elev** | 4464 |
| **trams** | 0 |
| **fastSixes** | 0 |
| **fastQuads** | 3 |
| **quad** | 2 |
| **triple** | 6 |
| **double** | 0 |
| **surface** | 3 |
| **total\_chairs** | 14 |
| **Runs** | 105.0 |
| **TerrainParks** | 4.0 |
| **LongestRun\_mi** | 3.3 |
| **SkiableTerrain\_ac** | 3000.0 |
| **Snow Making\_ac** | 600.0 |
| **daysOpenLastYear** | 123.0 |
| **yearsOpen** | 72.0 |
| **averageSnowfall** | 333.0 |
| **AdultWeekend** | 81.0 |
| **projectedDaysOpen** | 123.0 |
| **NightSkiing\_ac** | 600.0 |
| **resorts\_per\_state** | 12 |
| **resorts\_per\_100kcapita** | 1.122778 |
| **resorts\_per\_100ksq\_mile** | 8.161045 |
| **resort\_skiable\_area\_ac\_state\_ratio** | 0.140121 |
| **resort\_days\_open\_state\_ratio** | 0.129338 |
| **resort\_terrain\_park\_state\_ratio** | 0.148148 |
| **resort\_night\_skiing\_state\_ratio** | 0.84507 |
| **total\_chairs\_runs\_ratio** | 0.133333 |
| **total\_chairs\_skiable\_ratio** | 0.004667 |
| **fastQuads\_runs\_ratio** | 0.028571 |
| **fastQuads\_skiable\_ratio** | 0.001 |

## 5.6 Refit Model On All Available Data (excluding Big Mountain)

This next step requires some careful thought. We want to refit the model using all available data. But should we include Big Mountain data? On the one hand, we are not trying to estimate model performance on a previously unseen data sample, so theoretically including Big Mountain data should be fine. One might first think that including Big Mountain in the model training would, if anything, improve model performance in predicting Big Mountain's ticket price. But here's where our business context comes in. The motivation for this entire project is based on the sense that Big Mountain needs to adjust its pricing. One way to phrase this problem: we want to train a model to predict Big Mountain's ticket price based on data from all the other resorts! We don't want Big Mountain's current price to bias this. We want to calculate a price based only on its competitors.



X = ski\_data.loc[ski\_data.Name != "Big Mountain Resort", model.X\_columns]

y = ski\_data.loc[ski\_data.Name != "Big Mountain Resort", 'AdultWeekend']



len(X), len(y)

(276, 276)



model.fit(X, y)

Pipeline(steps=[('simpleimputer', SimpleImputer(strategy='median')),

('standardscaler', None),

('randomforestregressor',

RandomForestRegressor(n\_estimators=69, random\_state=47))])



cv\_results = cross\_validate(model, X, y, scoring='neg\_mean\_absolute\_error', cv=5, n\_jobs=-1)



cv\_results['test\_score']

array([-12.09690217, -9.30247694, -11.41595784, -8.10096706,

-11.04942819])



mae\_mean, mae\_std = np.mean(-1 \* cv\_results['test\_score']), np.std(-1 \* cv\_results['test\_score'])

mae\_mean, mae\_std

(10.393146442687748, 1.4712769116280346)

These numbers will inevitably be different to those in the previous step that used a different training data set. They should, however, be consistent. It's important to appreciate that estimates of model performance are subject to the noise and uncertainty of data!

## 5.7 Calculate Expected Big Mountain Ticket Price From The Model



X\_bm = ski\_data.loc[ski\_data.Name == "Big Mountain Resort", model.X\_columns]

y\_bm = ski\_data.loc[ski\_data.Name == "Big Mountain Resort", 'AdultWeekend']



bm\_pred = model.predict(X\_bm).item()



y\_bm = y\_bm.values.item()



print(f'Big Mountain Resort modelled price is ${bm\_pred:.2f}, actual price is ${y\_bm:.2f}.')

print(f'Even with the expected mean absolute error of ${mae\_mean:.2f}, this suggests there is room for an increase.')

Big Mountain Resort modelled price is $95.87, actual price is $81.00.

Even with the expected mean absolute error of $10.39, this suggests there is room for an increase.

This result should be looked at optimistically and doubtfully! The validity of our model lies in the assumption that other resorts accurately set their prices according to what the market (the ticket-buying public) supports. The fact that our resort seems to be charging that much less that what's predicted suggests our resort might be undercharging. But if ours is mispricing itself, are others? It's reasonable to expect that some resorts will be "overpriced" and some "underpriced." Or if resorts are pretty good at pricing strategies, it could be that our model is simply lacking some key data? Certainly we know nothing about operating costs, for example, and they would surely help.

## 5.8 Big Mountain Resort In Market Context

Features that came up as important in the modeling (not just our final, random forest model) included:

* vertical\_drop
* Snow Making\_ac
* total\_chairs
* fastQuads
* Runs
* LongestRun\_mi
* trams
* SkiableTerrain\_ac

A handy glossary of skiing terms can be found on the [ski.com](https://www.ski.com/ski-glossary) site. Some potentially relevant contextual information is that vertical drop, although nominally the height difference from the summit to the base, is generally taken from the highest [lift-served](http://verticalfeet.com/) point.

It's often useful to define custom functions for visualizing data in meaningful ways. The function below takes a feature name as an input and plots a histogram of the values of that feature. It then marks where Big Mountain sits in the distribution by marking Big Mountain's value with a vertical line using matplotlib's [axvline](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.axvline.html) function. It also performs a little cleaning up of missing values and adds descriptive labels and a title.



#Code task 1#

#Add code to the `plot\_compare` function that displays a vertical, dashed line

#on the histogram to indicate Big Mountain's position in the distribution

#Hint: plt.axvline() plots a vertical line, its position for 'feature1'

#would be `big\_mountain['feature1'].values, we'd like a red line, which can be

#specified with c='r', a dashed linestyle is produced by ls='--',

#and it's nice to give it a slightly reduced alpha value, such as 0.8.

#Don't forget to give it a useful label (e.g. 'Big Mountain') so it's listed

#in the legend.

def plot\_compare(feat\_name, description, state=None, figsize=(10, 5)):

"""Graphically compare distributions of features.

Plot histogram of values for all resorts and reference line to mark

Big Mountain's position.

Arguments:

feat\_name - the feature column name in the data

description - text description of the feature

state - select a specific state (None for all states)

figsize - (optional) figure size

"""

plt.subplots(figsize=figsize)

# quirk that hist sometimes objects to NaNs, sometimes doesn't

# filtering only for finite values tidies this up

if state is None:

ski\_x = ski\_data[feat\_name]

else:

ski\_x = ski\_data.loc[ski\_data.state == state, feat\_name]

ski\_x = ski\_x[np.isfinite(ski\_x)]

plt.hist(ski\_x, bins=30)

plt.axvline(x=big\_mountain[feat\_name].values, c='r', ls='--', alpha=0.8, label='Big Mountain')

plt.xlabel(description)

plt.ylabel('frequency')

plt.title(description + ' distribution for resorts in market share')

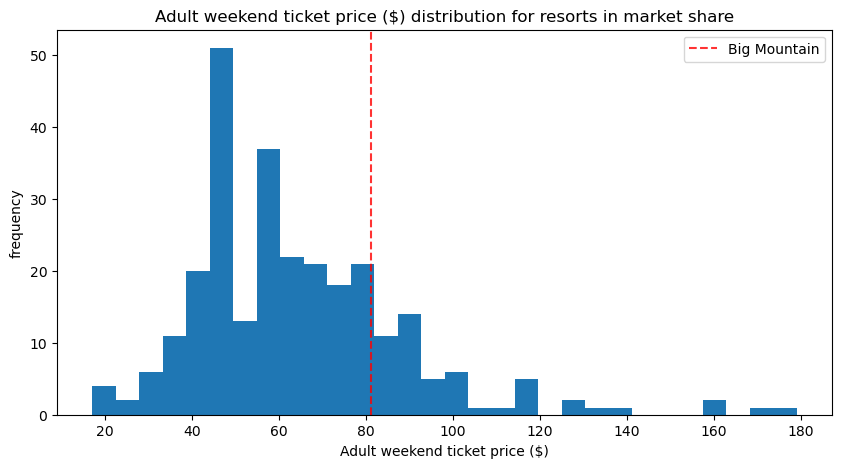
plt.legend()

### 5.8.1 Ticket price

Look at where Big Mountain sits overall amongst all resorts for price and for just other resorts in Montana.

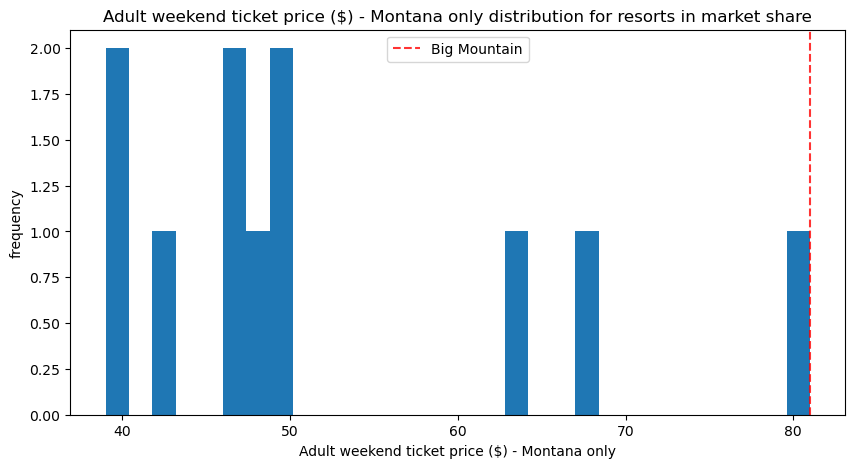


plot\_compare('AdultWeekend', 'Adult weekend ticket price ($)')





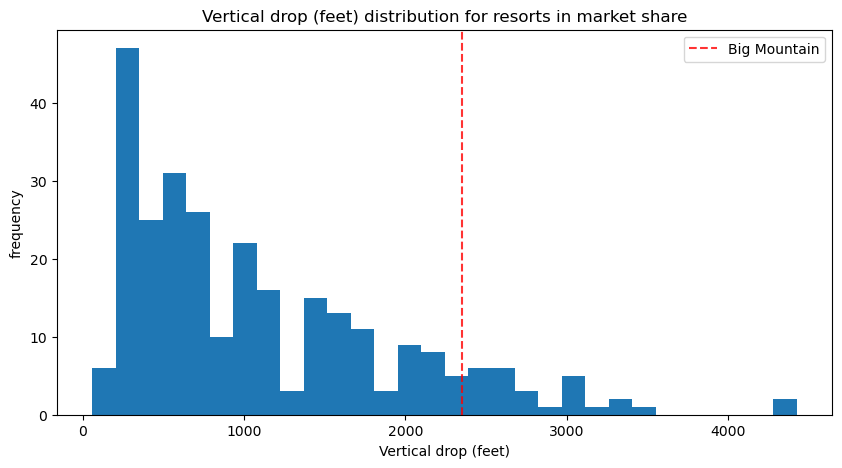
plot\_compare('AdultWeekend', 'Adult weekend ticket price ($) - Montana only', state='Montana')



### 5.8.2 Vertical drop



plot\_compare('vertical\_drop', 'Vertical drop (feet)')

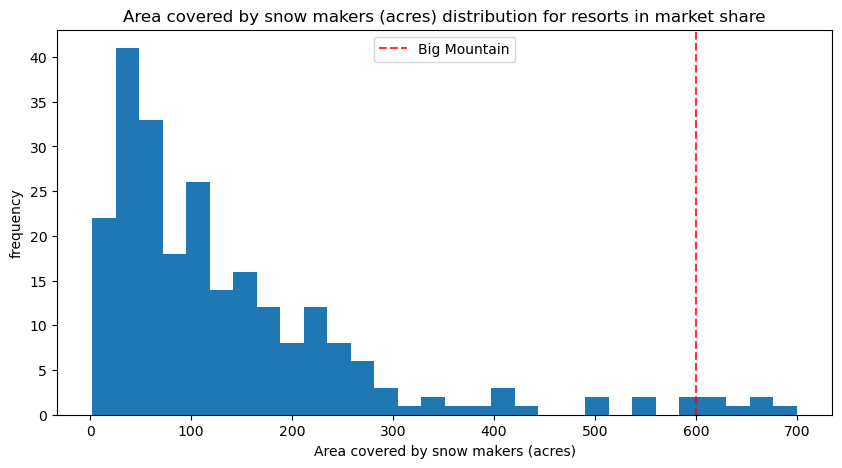


Big Mountain is doing well for vertical drop, but there are still quite a few resorts with a greater drop.

### 5.8.3 Snow making area



plot\_compare('Snow Making\_ac', 'Area covered by snow makers (acres)')

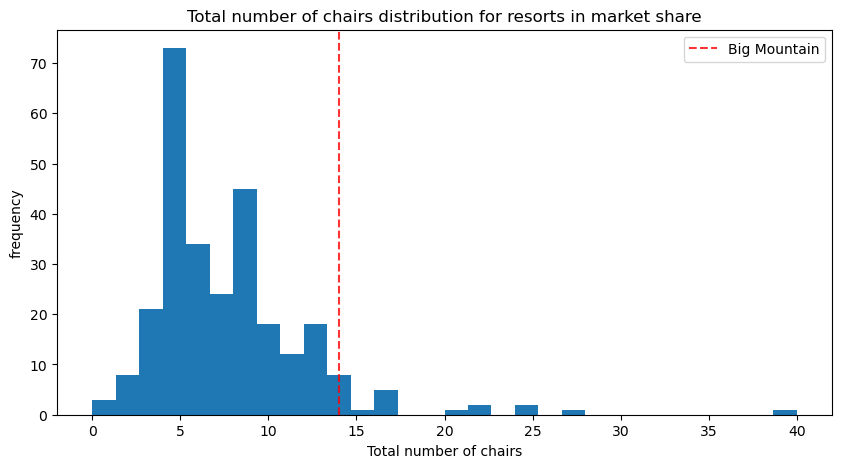


Big Mountain is very high up the league table of snow making area.

### 5.8.4 Total number of chairs



plot\_compare('total\_chairs', 'Total number of chairs')

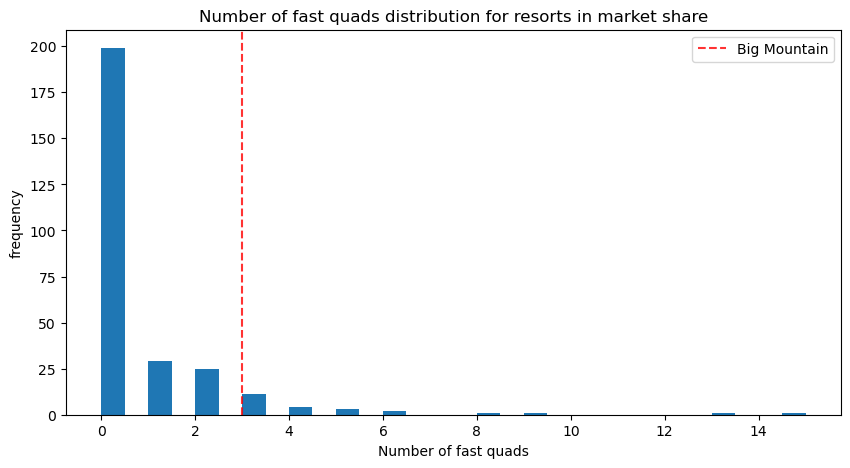


Big Mountain has amongst the highest number of total chairs, resorts with more appear to be outliers.

### 5.8.5 Fast quads



plot\_compare('fastQuads', 'Number of fast quads')

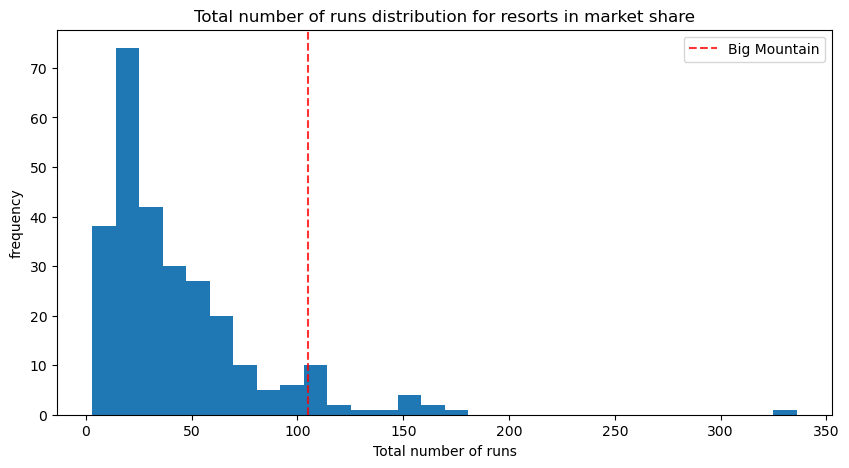


Most resorts have no fast quads. Big Mountain has 3, which puts it high up that league table. There are some values much higher, but they are rare.

### 5.8.6 Runs



plot\_compare('Runs', 'Total number of runs')

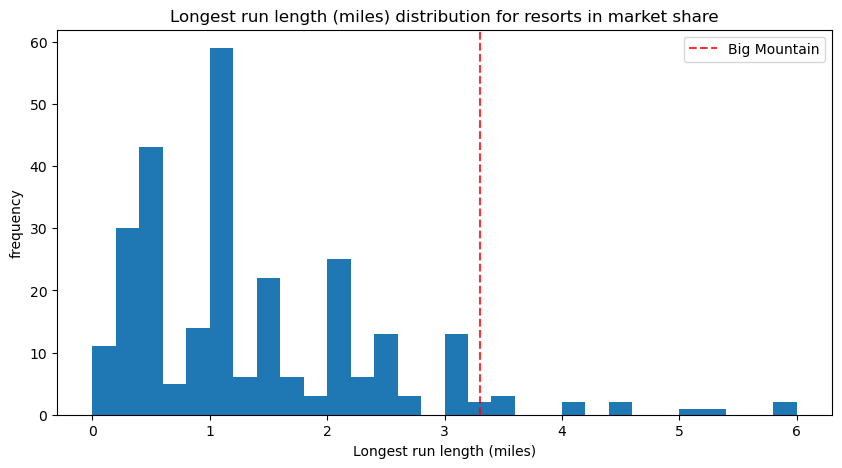


Big Mountain compares well for the number of runs. There are some resorts with more, but not many.

### 5.8.7 Longest run



plot\_compare('LongestRun\_mi', 'Longest run length (miles)')

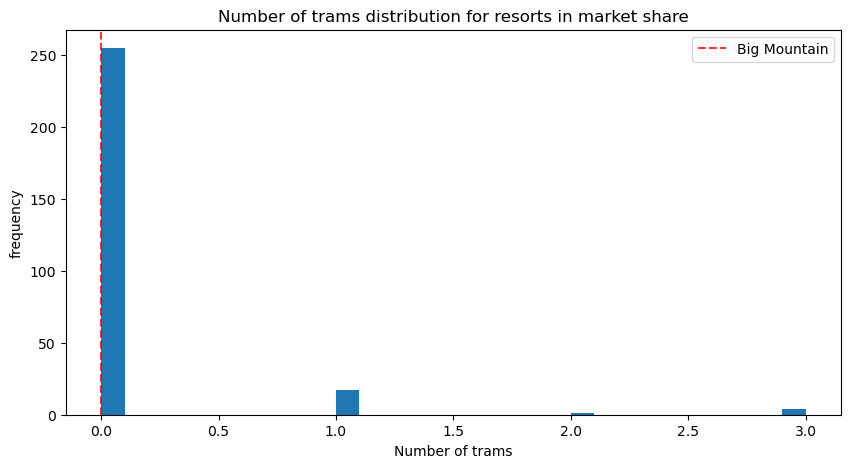


Big Mountain has one of the longest runs. Although it is just over half the length of the longest, the longer ones are rare.

### 5.8.8 Trams



plot\_compare('trams', 'Number of trams')

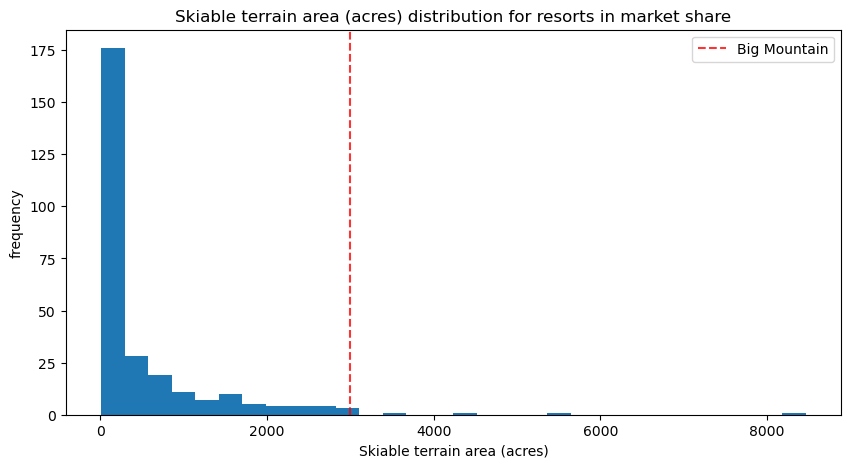


The vast majority of resorts, such as Big Mountain, have no trams.

### 5.8.9 Skiable terrain area



plot\_compare('SkiableTerrain\_ac', 'Skiable terrain area (acres)')



Big Mountain is amongst the resorts with the largest amount of skiable terrain.

## 5.9 Modeling scenarios

Big Mountain Resort has been reviewing potential scenarios for either cutting costs or increasing revenue (from ticket prices). Ticket price is not determined by any set of parameters; the resort is free to set whatever price it likes. However, the resort operates within a market where people pay more for certain facilities, and less for others. Being able to sense how facilities support a given ticket price is valuable business intelligence. This is where the utility of our model comes in.

The business has shortlisted some options:

1. Permanently closing down up to 10 of the least used runs. This doesn't impact any other resort statistics.
2. Increase the vertical drop by adding a run to a point 150 feet lower down but requiring the installation of an additional chair lift to bring skiers back up, without additional snow making coverage
3. Same as number 2, but adding 2 acres of snow making cover
4. Increase the longest run by 0.2 mile to boast 3.5 miles length, requiring an additional snow making coverage of 4 acres

The expected number of visitors over the season is 350,000 and, on average, visitors ski for five days. Assume the provided data includes the additional lift that Big Mountain recently installed.



expected\_visitors = 350\_000



all\_feats = ['vertical\_drop', 'Snow Making\_ac', 'total\_chairs', 'fastQuads',

'Runs', 'LongestRun\_mi', 'trams', 'SkiableTerrain\_ac']

big\_mountain[all\_feats]

|  | **vertical\_drop** | **Snow Making\_ac** | **total\_chairs** | **fastQuads** | **Runs** | **LongestRun\_mi** | **trams** | **SkiableTerrain\_ac** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **124** | 2353 | 600.0 | 14 | 3 | 105.0 | 3.3 | 0 | 3000.0 |



#Code task 2#

#In this function, copy the Big Mountain data into a new data frame

#(Note we use .copy()!)

#And then for each feature, and each of its deltas (changes from the original),

#create the modified scenario dataframe (bm2) and make a ticket price prediction

#for it. The difference between the scenario's prediction and the current

#prediction is then calculated and returned.

#Complete the code to increment each feature by the associated delta

def predict\_increase(features, deltas):

"""Increase in modelled ticket price by applying delta to feature.

Arguments:

features - list, names of the features in the ski\_data dataframe to change

deltas - list, the amounts by which to increase the values of the features

Outputs:

Amount of increase in the predicted ticket price

"""

bm2 = X\_bm.copy()

for f, d in zip(features, deltas):

bm2[f] += d

return model.predict(bm2).item() - model.predict(X\_bm).item()

### 5.9.1 Scenario 1

Close up to 10 of the least used runs. The number of runs is the only parameter varying.



[i for i in range(-1, -11, -1)]

[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]



runs\_delta = [i for i in range(-1, -11, -1)]

price\_deltas = [predict\_increase(['Runs'], [delta]) for delta in runs\_delta]



price\_deltas

[0.0,

-0.4057971014492807,

-0.6666666666666714,

-0.6666666666666714,

-0.6666666666666714,

-1.2608695652173907,

-1.2608695652173907,

-1.2608695652173907,

-1.7101449275362341,

-1.8115942028985472]



#Code task 3#

#Create two plots, side by side, for the predicted ticket price change (delta) for each

#condition (number of runs closed) in the scenario and the associated predicted revenue

#change on the assumption that each of the expected visitors buys 5 tickets

#There are two things to do here:

#1 - use a list comprehension to create a list of the number of runs closed from `runs\_delta`

#2 - use a list comprehension to create a list of predicted revenue changes from `price\_deltas`

runs\_closed = [-1 \* i for i in runs\_delta] #1

fig, ax = plt.subplots(1, 2, figsize=(10, 5))

fig.subplots\_adjust(wspace=0.5)

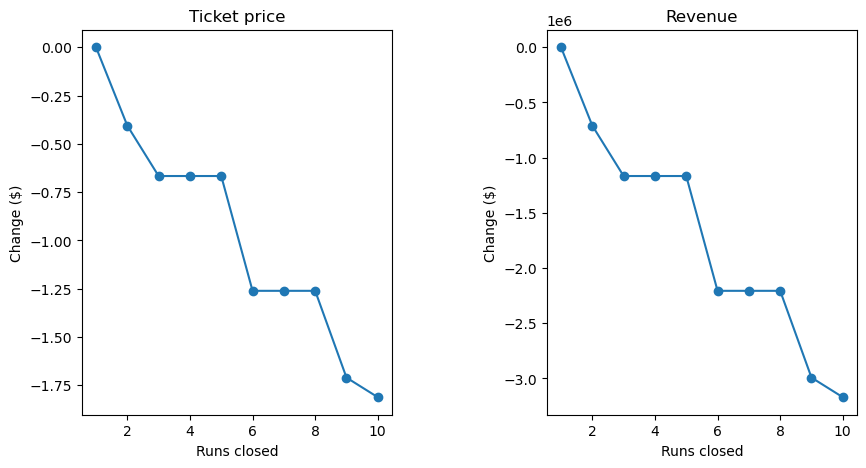
ax[0].plot(runs\_closed, price\_deltas, 'o-')

ax[0].set(xlabel='Runs closed', ylabel='Change ($)', title='Ticket price')

revenue\_deltas = [5 \* expected\_visitors \* i for i in price\_deltas] #2

ax[1].plot(runs\_closed, revenue\_deltas, 'o-')

ax[1].set(xlabel='Runs closed', ylabel='Change ($)', title='Revenue');



The model says closing one run makes no difference. Closing 2 and 3 successively reduces support for ticket price and so revenue. If Big Mountain closes down 3 runs, it seems they may as well close down 4 or 5 as there's no further loss in ticket price. Increasing the closures down to 6 or more leads to a large drop.

### 5.9.2 Scenario 2

In this scenario, Big Mountain is adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift.



#Code task 4#

#Call `predict\_increase` with a list of the features 'Runs', 'vertical\_drop', and 'total\_chairs'

#and associated deltas of 1, 150, and 1

ticket2\_increase = predict\_increase(['Runs', 'vertical\_drop', 'total\_chairs'], [1, 150, 1])

revenue2\_increase = 5 \* expected\_visitors \* ticket2\_increase



print(f'This scenario increases support for ticket price by ${ticket2\_increase:.2f}')

print(f'Over the season, this could be expected to amount to ${revenue2\_increase:.0f}')

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

### 5.9.3 Scenario 3

In this scenario, you are repeating the previous one but adding 2 acres of snow making.



#Code task 5#

#Repeat scenario 2 conditions, but add an increase of 2 to `Snow Making\_ac`

ticket3\_increase = predict\_increase(['Runs', 'vertical\_drop', 'total\_chairs', 'Snow Making\_ac'], [1, 150, 1, 2])

revenue3\_increase = 5 \* expected\_visitors \* ticket3\_increase



print(f'This scenario increases support for ticket price by ${ticket3\_increase:.2f}')

print(f'Over the season, this could be expected to amount to ${revenue3\_increase:.0f}')

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638



Such a small increase in the snow making area makes no difference!

### 5.9.4 Scenario 4

This scenario calls for increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability.



#Code task 6#

#Predict the increase from adding 0.2 miles to `LongestRun\_mi` and 4 to `Snow Making\_ac`

predict\_increase(['LongestRun\_mi', 'Snow Making\_ac'], [0.2, 4])

0.0

No difference whatsoever. Although the longest run feature was used in the linear model, the random forest model (the one we chose because of its better performance) only has longest run way down in the feature importance list.

## 5.10 Summary

**Q: 1** Write a summary of the results of modeling these scenarios. Start by starting the current position; how much does Big Mountain currently charge? What does your modelling suggest for a ticket price that could be supported in the marketplace by Big Mountain's facilities? How would you approach suggesting such a change to the business leadership? Discuss the additional operating cost of the new chair lift per ticket (on the basis of each visitor on average buying 5 day tickets) in the context of raising prices to cover this. For future improvements, state which, if any, of the modeled scenarios you'd recommend for further consideration. Suggest how the business might test, and progress, with any run closures.

**A: 1** The Big Mountain currently charges adult weekend price for $81.

This project is based on Big Mountain wanting to adjust its pricing. The problem is addressed by training a model to predict Big Mountain's ticket price based on data from all the other resorts. We don't want Big Mountain's current price to bias this, a price was calculated based only on its competitors.

Model was fitted using the pipeline, steps that imputes the median, random forest regressor, cross validate and results were MAE mean of 10.393, and MAE std of 1.471

These numbers will inevitably be different to those in the previous step that used a different training data set. They should, however, be consistent. It's important to appreciate that estimates of model performance are subject to the noise and uncertainty of data.

Next, calculating expected Big Mountain ticket price from the model and this was performed using the model predict method. Big Mountain Resort modelled price is $95.87, actual price is $81.00. Even with the expected mean absolute error of $10.39, this suggests there is room for an increase.

This result should be looked at optimistically and cautiously. The validity of our model lies in the assumption that other resorts accurately set their prices according to what the market (the ticket-buying public) supports. The fact that our resort seems to be charging that much less than what's predicted suggests our resort might be undercharging. The question is, are the other resorts also mispricing or do they have good pricing strategies? It's reasonable to expect that some resorts will be "overpriced" and some "underpriced." It could be that our model is simply lacking some key data. Knowing about operating costs might help.

From the modelling data, features that came up as important not just our final, random forest model, include vertical drop, snow making in acres, total chairs, fast quads, runs, longest run in miles and trams.

* In the US market, it looks like Montana (Big Mountain) is in the upper half in resort prices.
* In Montana (Big Mountain) is the most expensive resort price at $81.
* Big Mountain is doing well for vertical drop(2353 ft), but there are still quite a few resorts with a greater drop.
* Big Mountain is very high up the league table of snow making area (600 acres) among distribution for resorts in market share.
* Big Mountain has amongst the highest number of total chairs (14), resorts with more appear to be outliers.
* Most resorts have no fast quads. Big Mountain has 3, which puts it high up that league table. There are some values much higher, but they are rare.
* Big Mountain compares well for the number of runs (105). There are some resorts with more, but not many.
* Big Mountain has one of the longest runs (3.3 miles). Although it is just over half the length of the longest, the longer ones are rare.
* The vast majority of resorts, such as Big Mountain, have no trams.
* Big Mountain is amongst the resorts with the largest amount of skiable terrain (3000 acres).

The modelling suggests a ticket price $95.87 from actual price $81, there is room for increase that could be supported in the marketplace by Big Mountain based on the positive features/facilities that Big Mountain has to offer.

Big Mountain Resort has been reviewing potential scenarios for either cutting costs or increasing revenue (from ticket prices). Ticket price is not determined by any set of parameters; the resort is free to set whatever price it likes. However, the resort operates within a market where people pay more for certain facilities, and less for others. Being able to sense how facilities support a given ticket price is valuable business intelligence. This is where the utility of our model comes in.

The business has shortlisted some options:

* Permanently closing up to 10 of the least used runs. This doesn't impact any other resort statistics.
* Increase the vertical drop by adding a run to a point 150 feet lower down but requiring the installation of an additional chair lift to bring skiers back up, without additional snow making coverage. Making vertical drop 2503 ft.
* Same as number 2 (vertical drop to 2503 ft), but adding 2 acres of snow making cover. Making it 602 acres.
* Increase the longest run by 0.2 mile to boast 3.5 miles length, requiring an additional snow making coverage of 4 acres

**Testing**

The following prediction tests were done, taking into account the following:

The expected number of visitors over the season is 350,000 and, on average, visitors ski for five days. Assume the provided data includes the additional lift that Big Mountain recently installed.

Increase in modelled ticket price applying the amounts by which to increase the values of the features. This will output the amount in increase in the predicted price.

**Scenario 1**

Close 1 of the least used runs. The number of runs is the only parameter varying.

Price deltas results: Negative result for all features means people would want to decrease the price when closing more than one run.

The model says closing one run makes no difference in the price. Closing 2 or more reduces support for ticket price and revenue. The model shows a couple of plateaus in the closures. Closing 3 has the same effect as closing 4 or 5. But the operational cost savings may offset the loss in ticket prices of $0.67. We also see a large drop to the next plateau of 6, 7, or 8 closures of $1.26 in price reduction.

**Scenario 2**

In this scenario, Big Mountain is adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift.

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

The question to be investigated is, does the added revenue cover the expenses of extending a run by the 150 ft drop, including possible new lift to the new starting point. Also, is there a run that is extendable with property and geographic limitations within the overall property?

**Scenario 3**

In this scenario, you are repeating the previous one but adding 2 acres of snow making.

This scenario increases support for ticket price by $1.99

Over the season, this could be expected to amount to $3474638

Such a small increase in the snow making area makes no difference.

**Scenario 4**

This scenario calls for increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability.

Predict increase was 0. There is no effect on ticket price or revenue.

No difference whatsoever. Although the longest run feature was used in the linear model, the random forest model (the one we chose because of its better performance) only has longest runway down in the feature importance list.

**Conclusion**

Based on the modeled scenarios recommended for further consideration, can be summarized as follows:

Overall, closing 1 run will not dramatically affect ticket price or revenue. Adding a run, increasing the vertical drop by 150 ft and an additional chair lift will support increasing the ticket price by $1.99 and having a predicted revenue of $3474638 over a season. Adding a run, increasing the vertical drop by 150 ft, possibly adding a chair lift, and adding 2 acres of snow making area resulted in supporting increase in ticket price ($1.99) and predicted revenues per season of $3474638 but the addition of the 2 acres snow making area does not make a difference. Lastly, adding 0.2 miles to the longest run and guaranteeing snow coverage by adding 4 acres of snow making area did not make a difference.

I would recommend a combination of scenario 1 and 2. Scenario 1 tells us that by closing one run will not affect the ticket price or revenue. But losing one run may also have operational cost savings that can be applied to the bottom line. Modifying a run when considering scenario 2 to adding a vertical drop of 150 ft. This indicates support for an increase in ticket prices and revenues. This scenario will also entail additional investigation into the cost of adding more vertical drop, possible redirection of a run and answer the question of the feasibility of physically adding the drop needs to be investigated.

Maybe further testing additional run closures against the other features in the training and test data plus the predicted number of visitors will show us more information on which additional features will support the ticket price and revenue increase.

## 5.11 Further work

**Q: 2** What next? Highlight any deficiencies in the data that hampered or limited this work. The only price data in our dataset were ticket prices. You were provided with information about the additional operating cost of the new chair lift, but what other cost information would be useful? Big Mountain was already fairly high on some of the league charts of facilities offered, but why was its modeled price so much higher than its current price? Would this mismatch come as a surprise to the business executives? How would you find out? Assuming the business leaders felt this model was useful, how would the business make use of it? Would you expect them to come to you every time they wanted to test a new combination of parameters in a scenario? We hope you would have better things to do, so how might this model be made available for business analysts to use and explore?

**A: 2**

We only have data calculations of ticket prices based only on the competitors of Big Mountain resort with the assumption that prices are set according to what the market supports. According to the model, our resort might be charging less than predicted. The question is, is the model we are using lacking any key data, like operating costs of the resorts? This might be something to be investigated further, which may have limited this model.

Aside from the operating cost of the additional chair lift as mentioned, another useful piece of data would be operating cost for the other facilities/features that this resort has to offer. This would include overall operating cost, source of general revenues of the resort and which facilities were doing well or not.

Based on the data, the resort is basing their ticket price according to what the market will support in their area.

Asking the main stakeholders as to what they based their original resort ticket prices might be a way of finding out the reasons why the business executives chose the ticket pricing that they have.

If this model is useful to the business, then they could apply the increase in price as recommended by the data. Any new parameters that are related to the present business problem may be tested but if parameters that are not directly related may be reviewed if it makes sense to add it or make a new project to investigate further.

It would be better to make recommendations based on parameters that the business leaders may have used in the past, to base any future business decisions.

This information could be explored and used by business analysts through Kaggle with using a different name if this is allowed by the business.