5 Modeling

5.1 Contents

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

In [12]:
```
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

In [13]:
```
# This isn't exactly production-grade, but a quick check for development
# These checks can save some head-scratching in development when moving fro
# 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

In [14]:
```
path = "/Users/jasonzhou/Documents/GitHub/DataScienceGuidedCapstone"
os.chdir(path)
```

In [15]:
```
ski_data = pd.read_csv('data/ski_data_step3_features.csv')
```

In [16]:
```
big_mountain = ski_data[ski_data.Name == 'Big Mountain Resort']
```

In [17]:
```
big_mountain.T
```

Out[17]:

	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

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.

In [18]:
```
X = ski_data.loc[ski_data.Name != "Big Mountain Resort", model.X_columns]
y = ski_data.loc[ski_data.Name != "Big Mountain Resort", 'AdultWeekend']
```

In [19]:
```
len(X), len(y)
```

Out[19]:
```
(276, 276)
```

In [20]:
```
model.fit(X, y)
```

Out[20]:
```
Pipeline(memory=None,
         steps=[('simpleimputer',
                 SimpleImputer(add_indicator=False, copy=True, fill_value
=None,
                               missing_values=nan, strategy='median',
                               verbose=0)),
                ('standardscaler', None),
                ('randomforestregressor',
                 RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                       criterion='mse', max_depth=None,
                                       max_features='auto', max_leaf_node
s=None,
                                       max_samples=None,
                                       min_impurity_decrease=0.0,
                                       min_impurity_split=None,
                                       min_samples_leaf=1, min_samples_sp
lit=2,
                                       min_weight_fraction_leaf=0.0,
                                       n_estimators=69, n_jobs=None,
                                       oob_score=False, random_state=47,
                                       verbose=0, warm_start=False))],
         verbose=False)
```

In [21]:
```
cv_results = cross_validate(model, X, y, scoring='neg_mean_absolute_error',
```

In [22]:
```
cv_results['test_score']
```

Out[22]:
```
array([-12.10337215,  -9.28661397, -11.41279578,  -8.06408169,
       -11.05864559])
```

In [23]:
```
mae_mean, mae_std = np.mean(-1 * cv_results['test_score']), np.std(-1 * cv_
mae_mean, mae_std
```

Out[23]:
```
(10.3851018351214, 1.487015739861695)
```

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

In [24]:
```
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']
```

In [25]:
```
bm_pred = model.predict(X_bm).item()
```

In [26]:
```
y_bm = y_bm.values.item()
```

In [27]:
```
print(f'Big Mountain Resort modelled price is ${bm_pred:.2f}, actual price
print(f'Even with the expected mean absolute error of ${mae_mean:.2f}, this
```

```
Big Mountain Resort modelled price is $94.22, 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.

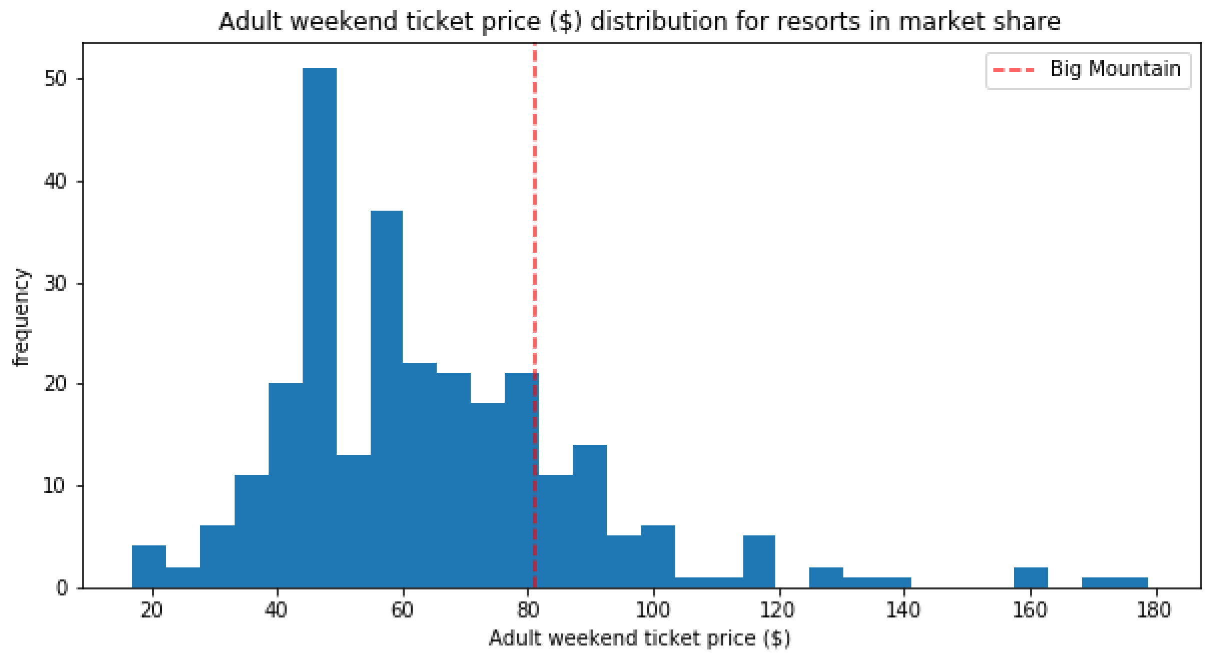
It's often useful to define custom functions for visualizing data in meaningful ways. The function below takes a feature name as an input and plots a histogram of the values of that feature. It then marks where Big Mountain sits in the distribution by marking Big Mountain's value with a vertical line 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.

In [28]:
```
#Code task 1#
#Add code to the `plot_compare` function that displays a vertical, dashed l
#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
#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 liste
#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
    plt.xlabel(description)
    plt.ylabel('frequency')
    plt.title(description + ' distribution for resorts in market share')
    plt.legend()
```

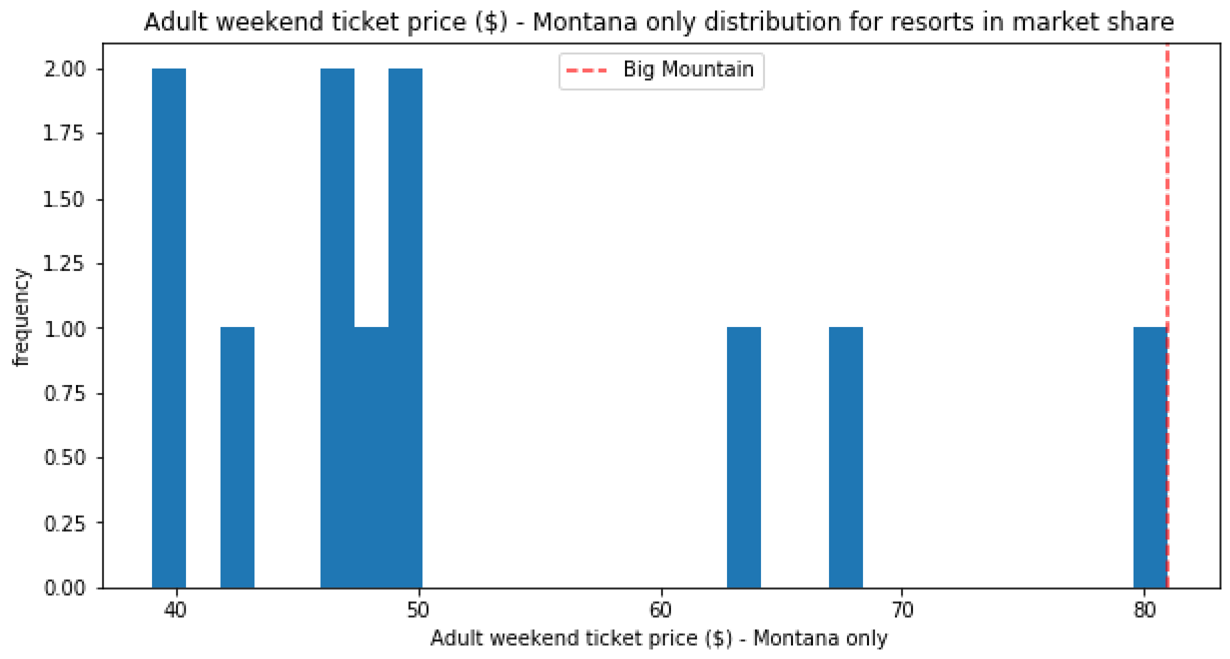
5.8.1 Ticket price

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

In [29]:
```
plot_compare('AdultWeekend', 'Adult weekend ticket price ($)')
```

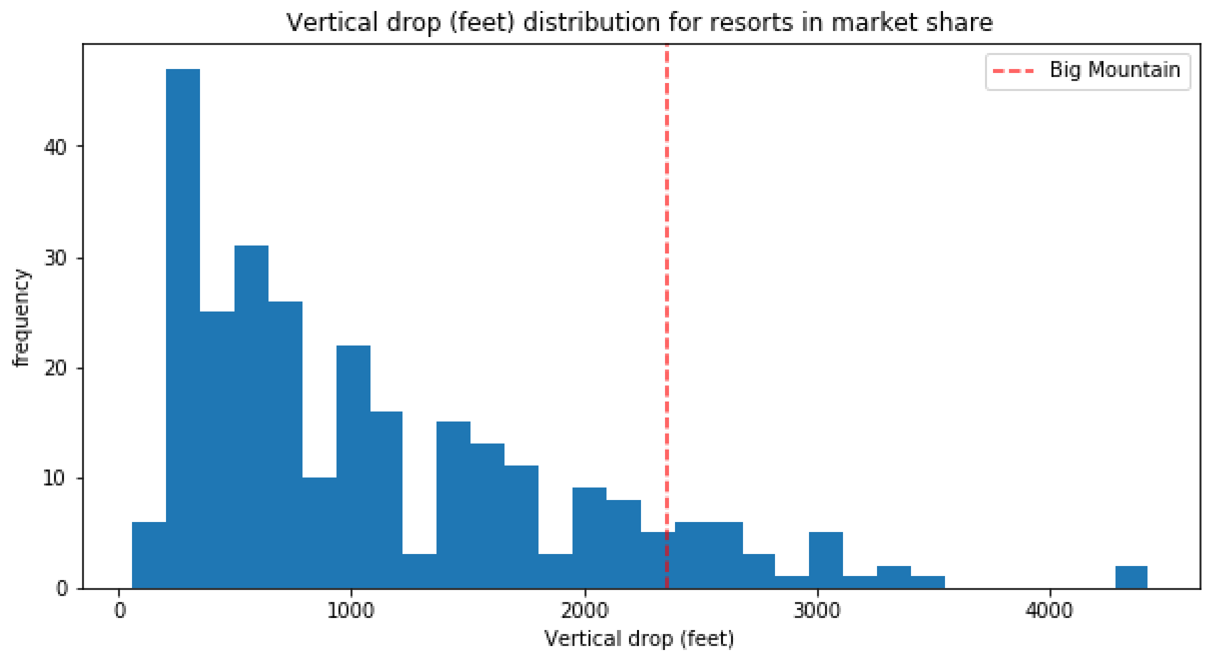


In [30]:
```
plot_compare('AdultWeekend', 'Adult weekend ticket price ($) - Montana only
```



5.8.2 Vertical drop

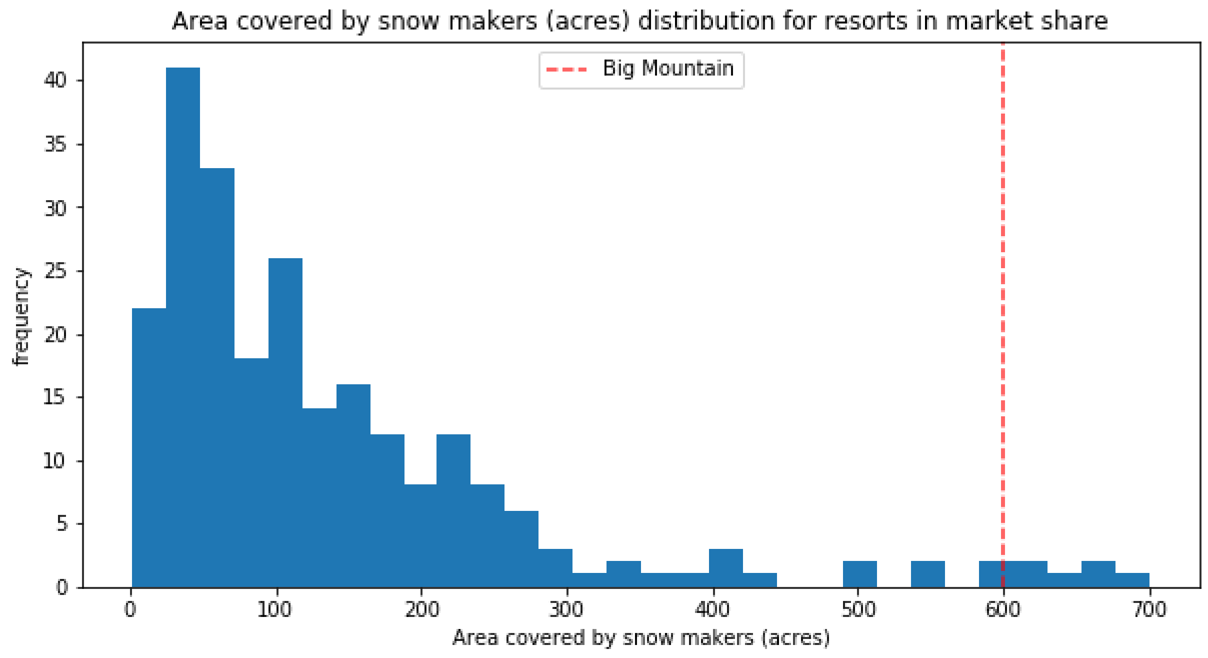
```
In [31]: plot_compare('vertical_drop', 'Vertical drop (feet)')
```



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

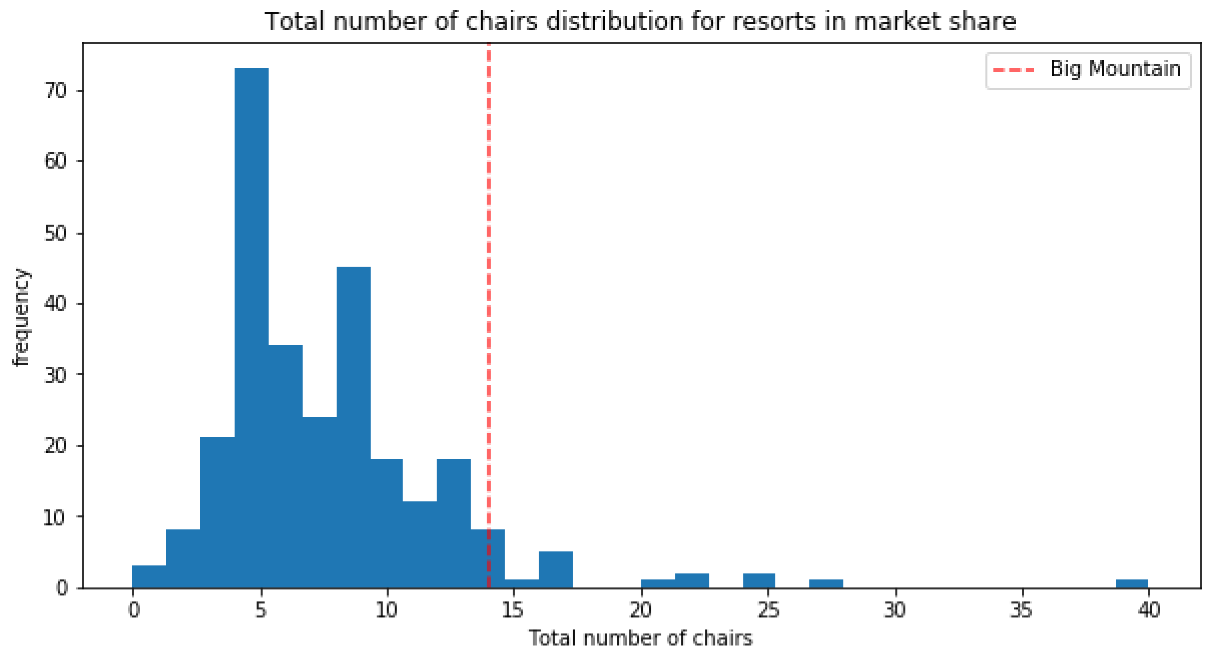
```
In [32]: plot_compare('Snow Making_ac', 'Area covered by snow makers (acres)')
```



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

5.8.4 Total number of chairs

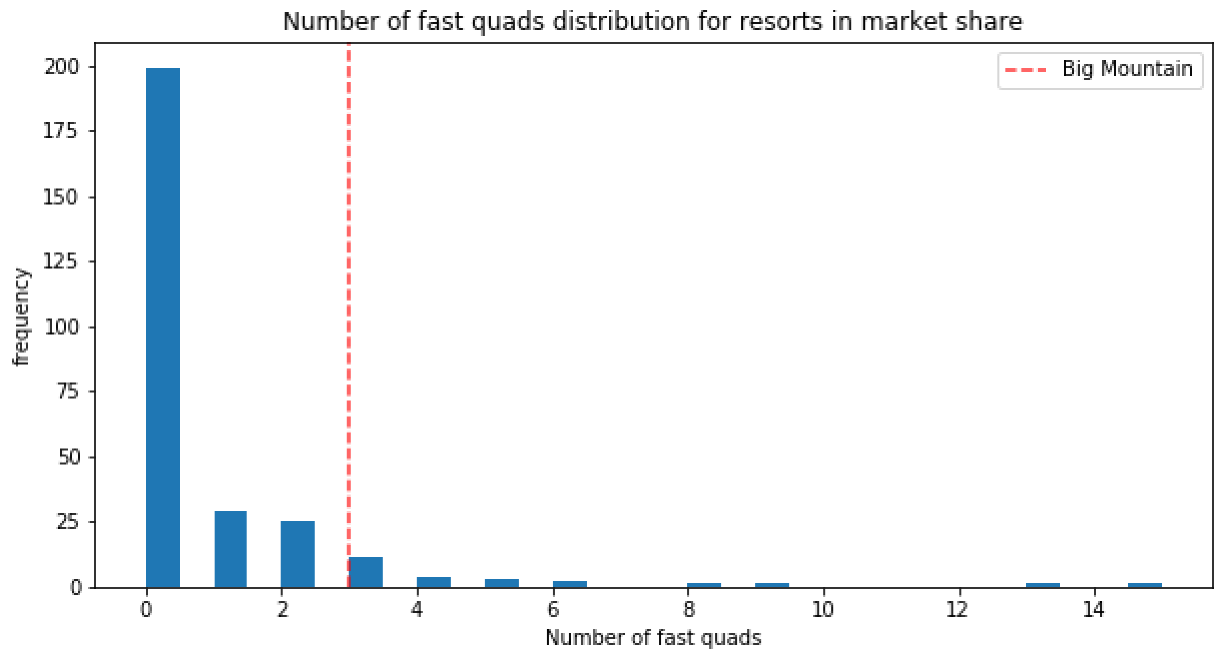
```
In [33]: plot_compare('total_chairs', 'Total number of chairs')
```



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

5.8.5 Fast quads

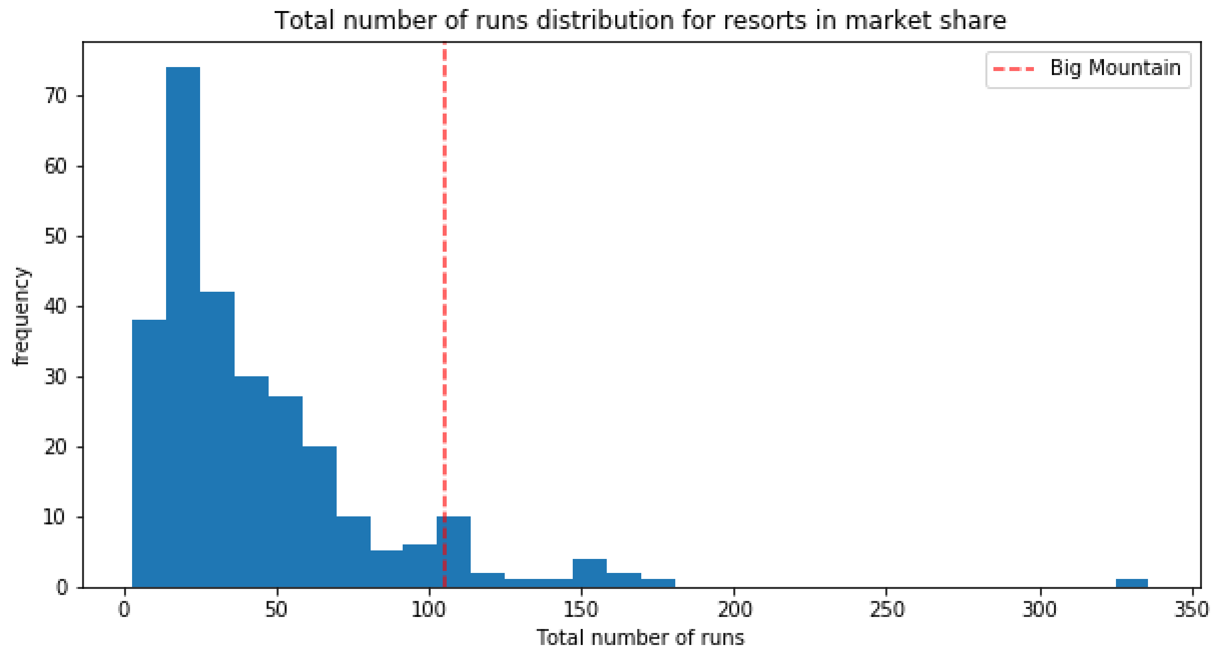
```
In [34]: plot_compare('fastQuads', 'Number of fast quads')
```



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

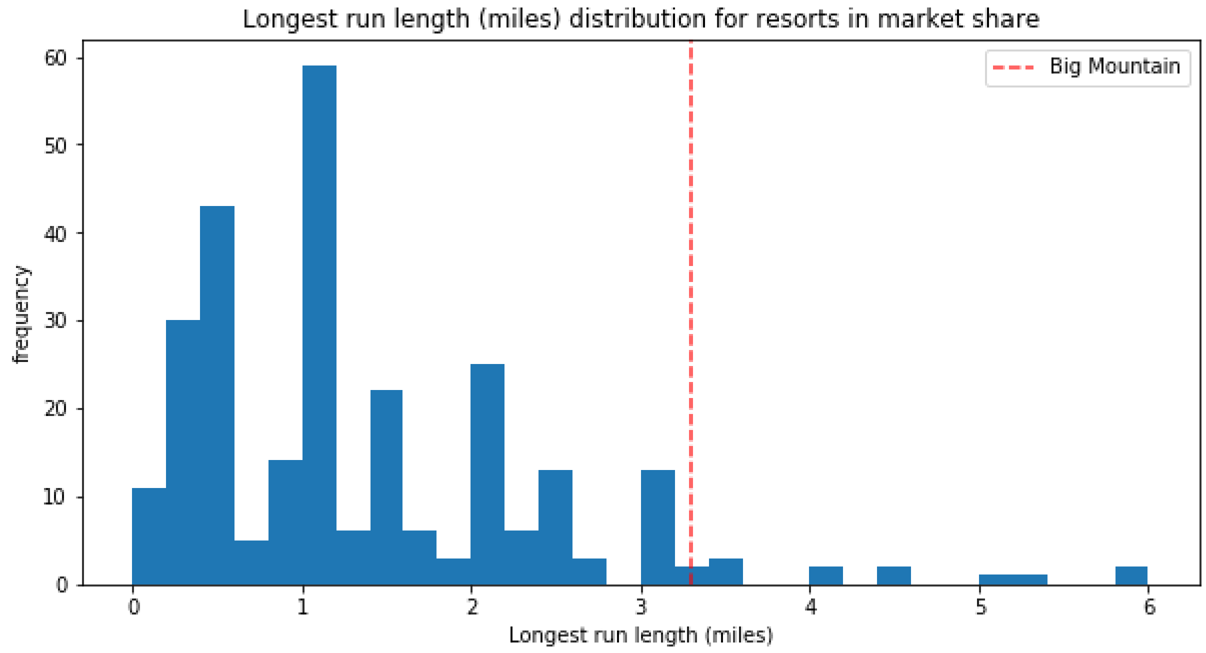
In [35]:
```
plot_compare('Runs', 'Total number of runs')
```



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

5.8.7 Longest run

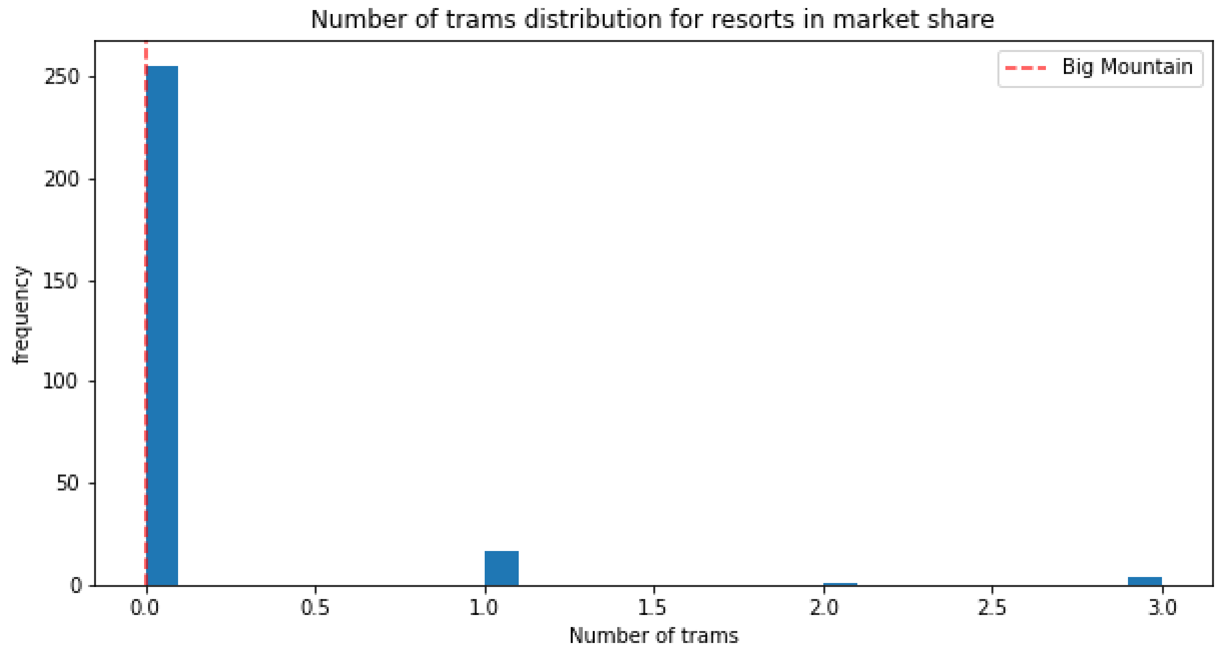
```
In [36]: plot_compare('LongestRun_mi', 'Longest run length (miles)')
```



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

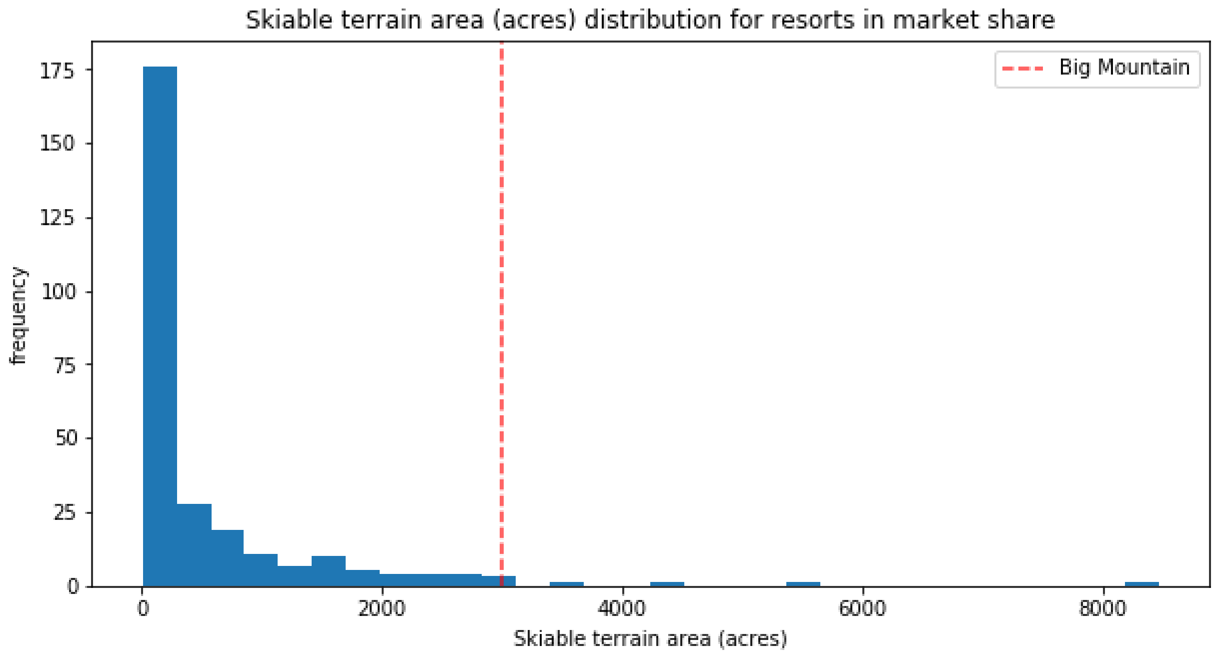
```
In [37]: plot_compare('trams', 'Number of trams')
```



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

5.8.9 Skiable terrain area

In [38]:
```
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.

In [39]:
```
expected_visitors = 350_000
```

In [40]:
```
all_feats = ['vertical_drop', 'Snow Making_ac', 'total_chairs', 'fastQuads'
             'Runs', 'LongestRun_mi', 'trams', 'SkiableTerrain_ac']
big_mountain[all_feats]
```

Out[40]:

	vertical_drop	Snow Making_ac	total_chairs	fastQuads	Runs	LongestRun_mi	trams	SkiableTerrain
124	2353	600.0	14	3	105.0	3.3	0	30(

In [41]:
```
#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 origin
#create the modified scenario dataframe (bm2) and make a ticket price predi
#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 cha
    deltas - list, the amounts by which to increase the values of the featu

    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.

In [42]:
```
[i for i in range(-1, -11, -1)]
```

Out[42]:
```
[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
```

In [43]:
```
runs_delta = [i for i in range(-1, -11, -1)]
price_deltas = [predict_increase(['Runs'], [delta]) for delta in runs_delta
```

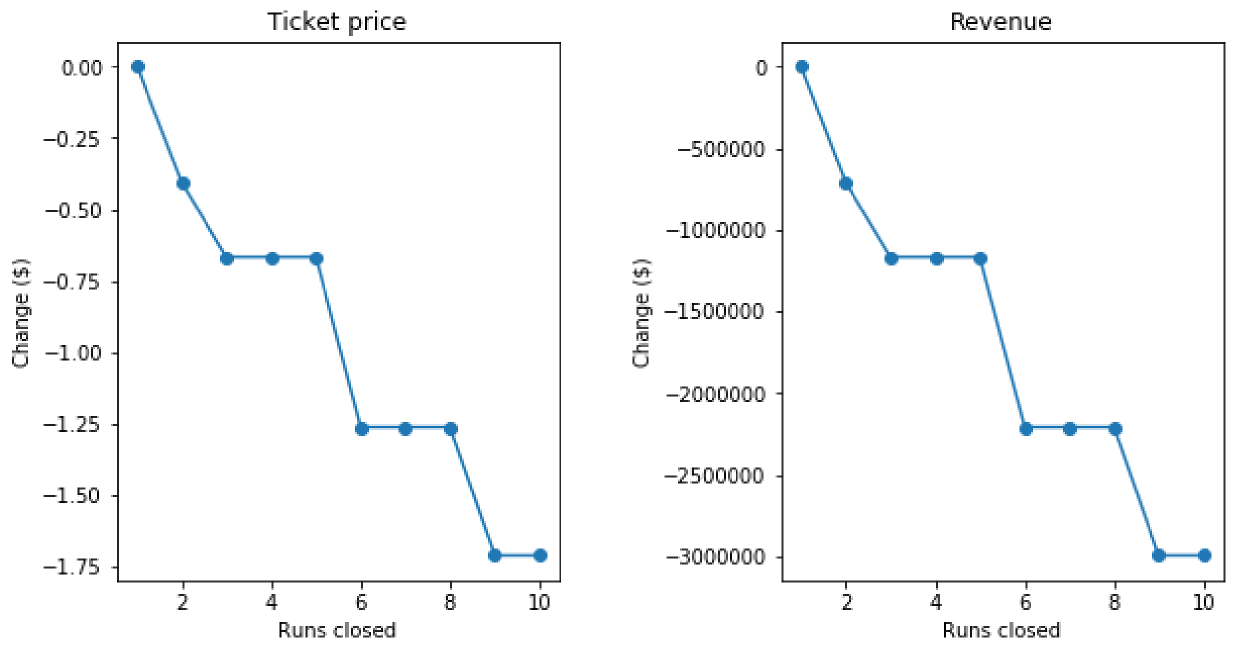
In [44]:
```
price_deltas
```

Out[44]:
```
[0.0,
 -0.4057971014492807,
 -0.6666666666666714,
 -0.6666666666666714,
 -0.6666666666666714,
 -1.2608695652173907,
 -1.2608695652173907,
 -1.2608695652173907,
 -1.7101449275362341,
 -1.7101449275362341]
```

In [45]:
```
runs_delta
```

Out[45]:
```
[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
```

In [46]:
```
#Code task 3#
#Create two plots, side by side, for the predicted ticket price change (del
#condition (number of runs closed) in the scenario and the associated predi
#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
#2 - use a list comprehension to create a list of predicted revenue changes
runs_closed = [-1 * k for k 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 ($)', title='Ticket price')
revenue_deltas = [5 * expected_visitors * j for j in price_deltas] #2
ax[1].plot(runs_closed, revenue_deltas, 'o-')
ax[1].set(xlabel='Runs closed', ylabel='Change ($)', 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.

In [47]:
```
#Code task 4#
#Call `predict_increase` with a list of the features 'Runs', 'vertical_drop
#and associated deltas of 1, 150, and 1
ticket2_increase = predict_increase(['Runs', 'vertical_drop', 'total_chairs
revenue2_increase = 5 * expected_visitors * ticket2_increase
```

In [48]:
```
print(f'This scenario increases support for ticket price by ${ticket2_incre
print(f'Over the season, this could be expected to amount to ${revenue2_inc
```

```
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.

In [50]:
```
ticket5_increase = predict_increase(['Runs'], [1])
print(ticket5_increase)
```

```
0.0
```

In [51]:
```
#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
revenue3_increase = 5 * expected_visitors * ticket3_increase
```

In [52]:
```
print(f'This scenario increases support for ticket price by ${ticket3_incre
print(f'Over the season, this could be expected to amount to ${revenue3_inc
```

```
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.

In [ ]:
```
#Code task 6#
#Predict the increase from adding 0.2 miles to `LongestRun_mi` and 4 to `Sn
predict_increase(['LongestRun_mi', 'Snow Making_ac'], [0.2, 4])
```

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?

A: 2 Your answer here