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 [7]:
```
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' - this line did not work as given ∴ edit b
expected_model_version = 1.0
model_path = 'Ajita_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("All Good")
else:
    print("Expected model not found")
```

```
All Good
```

5.5 Load Data

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

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

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

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_night_skiing_state_ratio	0.84507
resort_terrain_park_state_ratio	0.148148
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.

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(steps=[('simpleimputer', SimpleImputer(strategy='median')),
                ('standardscaler', None),
                ('randomforestregressor',
                 RandomForestRegressor(n_estimators=615, random_state=4
7))])
```

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

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

Out[22]:
```
array([-11.94936498,  -9.21277458, -11.43879468,  -7.81730229,
       -10.95806622])
```

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

Out[23]:
```
(10.275260550100306, 1.5357796818373215)
```

NB: 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 $92.77, actual price is $81.00.
Even with the expected mean absolute error of $10.28, 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 than 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.

In [65]:
```
# added revenue when using predicted price : $20597500.0
# 5*expected_visitors*11.77

# required price increase to cover cost of chair (1,540,000) : $0.88
# 1540000/(5*expected_visitors)

# required price increase for an increase in revenue of 20%
# 81*0.2 + 1540000/(5*expected_visitors) : 17.08

#revenue increase with $5 per ticket : %6
# 5/81
```

5.8 Big Mountain Resort In Market Context

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.

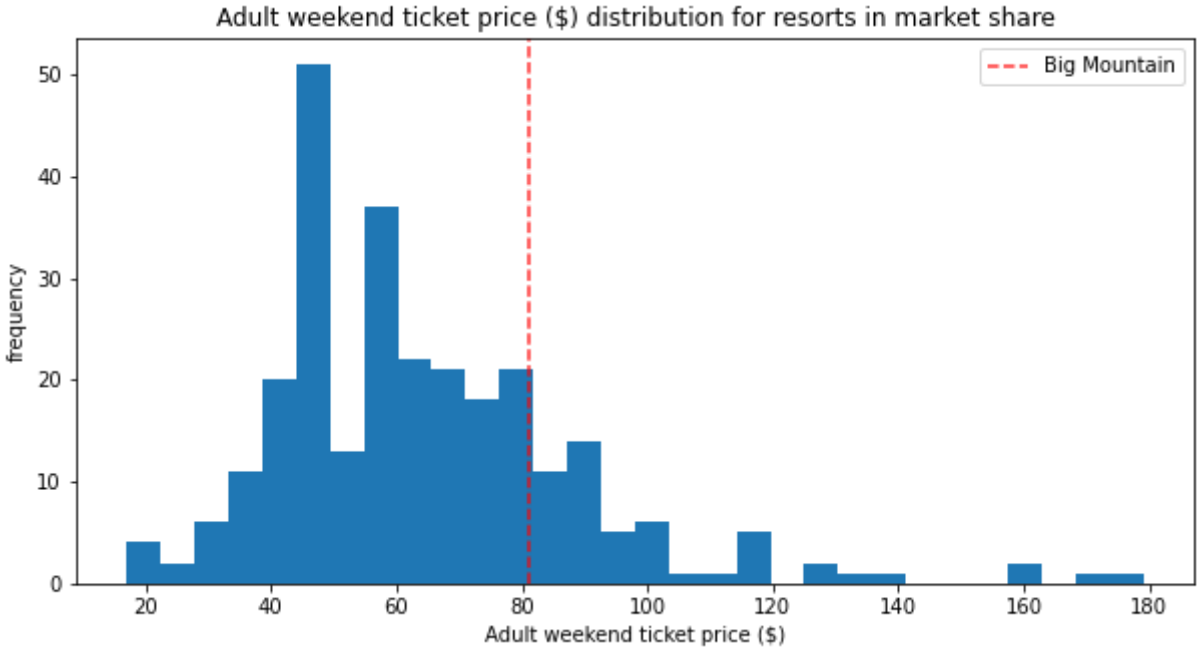
It's often useful to define custom functions for visualizing data in meaningful ways. The function below takes a feature name as an input and plots a histogram of the values of that feature. It then marks where Big Mountain sits in the distribution by marking Big Mountain's value with a vertical line 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]:
```
task 1#
ode to the `plot_compare` function that displays a vertical, dashed line
e histogram to indicate Big Mountain's position in the distribution
 plt.axvline() plots a vertical line, its position for 'feature1'
 be `big_mountain['feature1'].values, we'd like a red line, which can be
fied with c='r', a dashed linestyle is produced by ls='--',
t's nice to give it a slightly reduced alpha value, such as 0.8.
 forget to give it a useful label (e.g. 'Big Mountain') so it's listed
e legend.
ot_compare(feat_name, description, state=None, figsize=(10, 5)):
'Graphically compare distributions of features.

ot histogram of values for all resorts and reference line to mark
g Mountain's position.

guments:
at_name - the feature column name in the data
scription - text description of the feature
ate - select a specific state (None for all states)
gsize - (optional) figure size
'

t.subplots(figsize=figsize)
quirk that hist sometimes objects to NaNs, sometimes doesn't
filtering only for finite values tidies this up
 state is None:
  ski_x = ski_data[feat_name]
se:
  ski_x = ski_data.loc[ski_data.state == state, feat_name]
i_x = ski_x[np.isfinite(ski_x)]
t.hist(ski_x, bins=30)
t.axvline(x=big_mountain[feat_name].values, c='r', ls='--', alpha=0.8, label
t.xlabel(description)
t.ylabel('frequency')
t.title(description + ' distribution for resorts in market share')
t.legend()
```

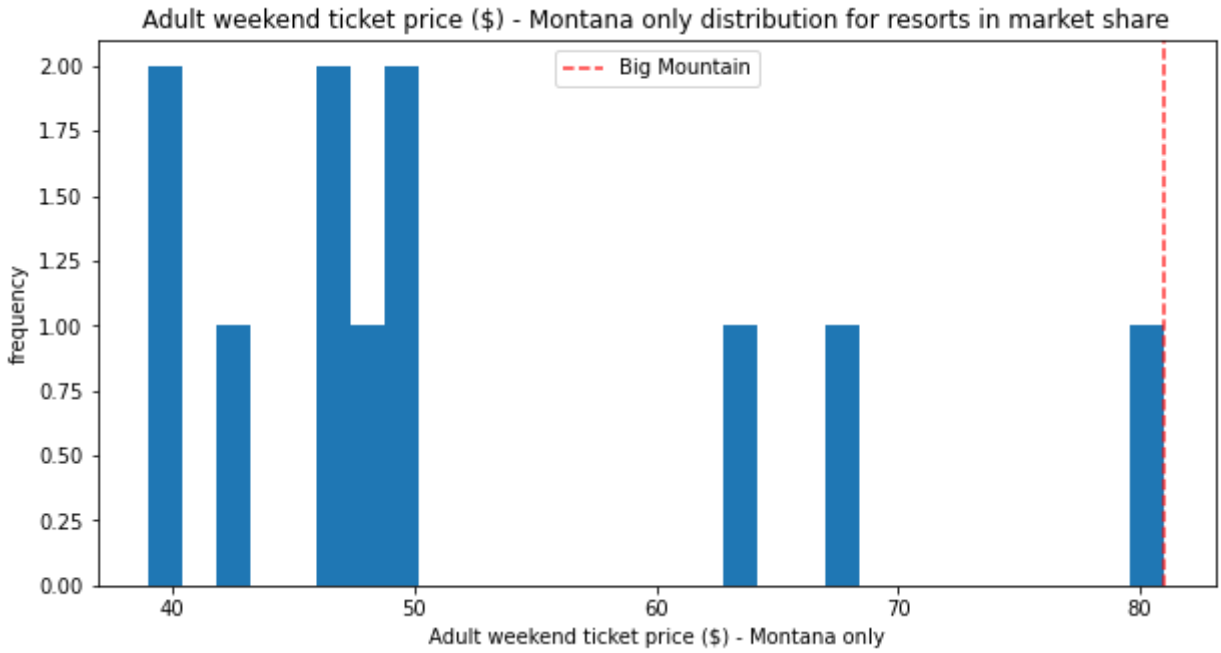
5.8.1 Ticket price

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

In [17]:
```
plot_compare('AdultWeekend', 'Adult weekend ticket price ($)')
```

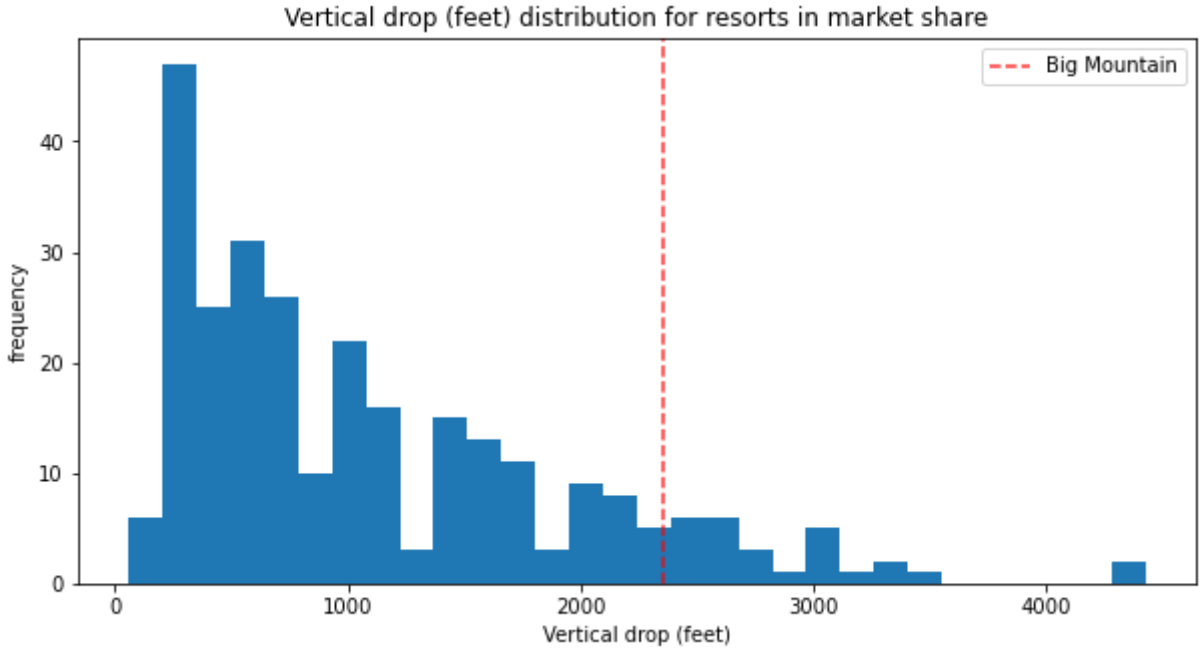


In [18]:
```
plot_compare('AdultWeekend', 'Adult weekend ticket price ($) - Montana only
```



5.8.2 Vertical drop

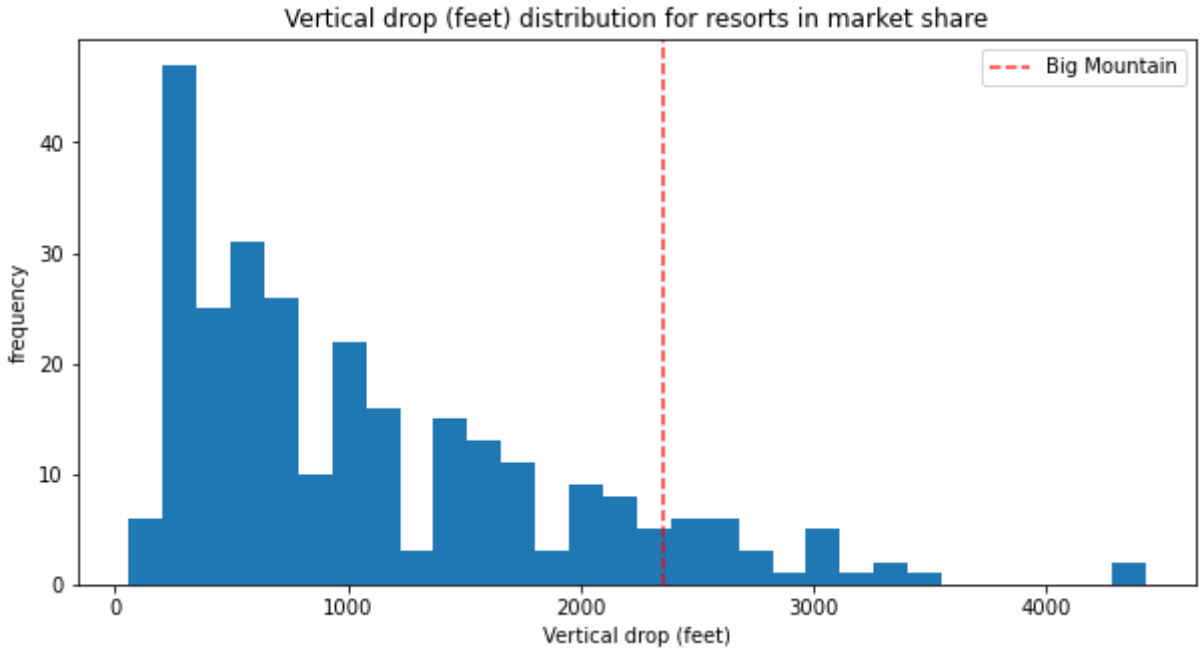
In [19]:
```
plot_compare('vertical_drop', 'Vertical drop (feet)')
```



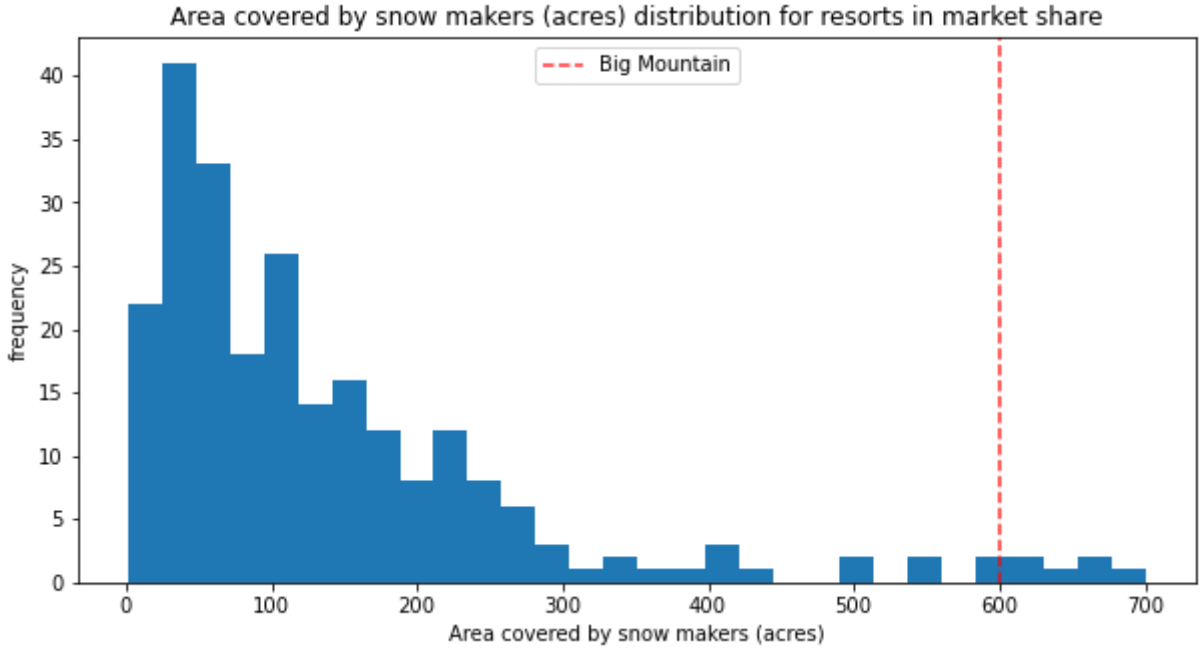
Big Mountain is doing well for vertical drop, but there are still quite a few resorts with a greater drop.

5.8.3 Snow making area

In [19]:
```
plot_compare('vertical_drop', 'Vertical drop (feet)')
```



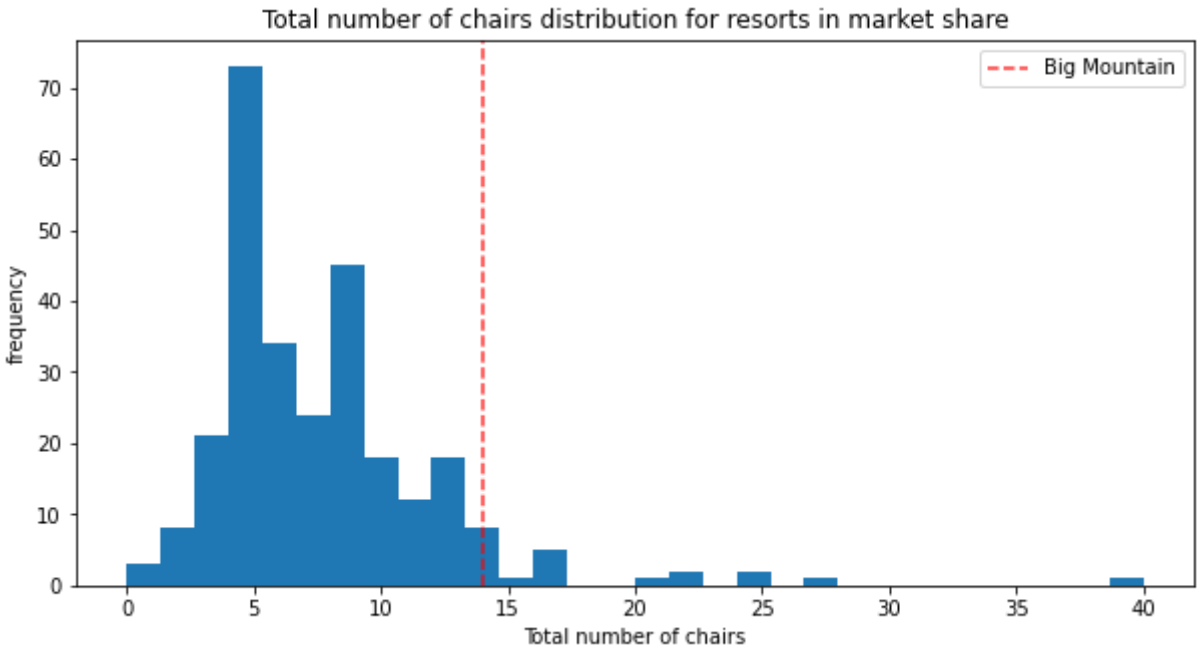
In [20]:
```
plot_compare('Snow Making_ac', 'Area covered by snow makers (acres)')
```



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

5.8.4 Total number of chairs

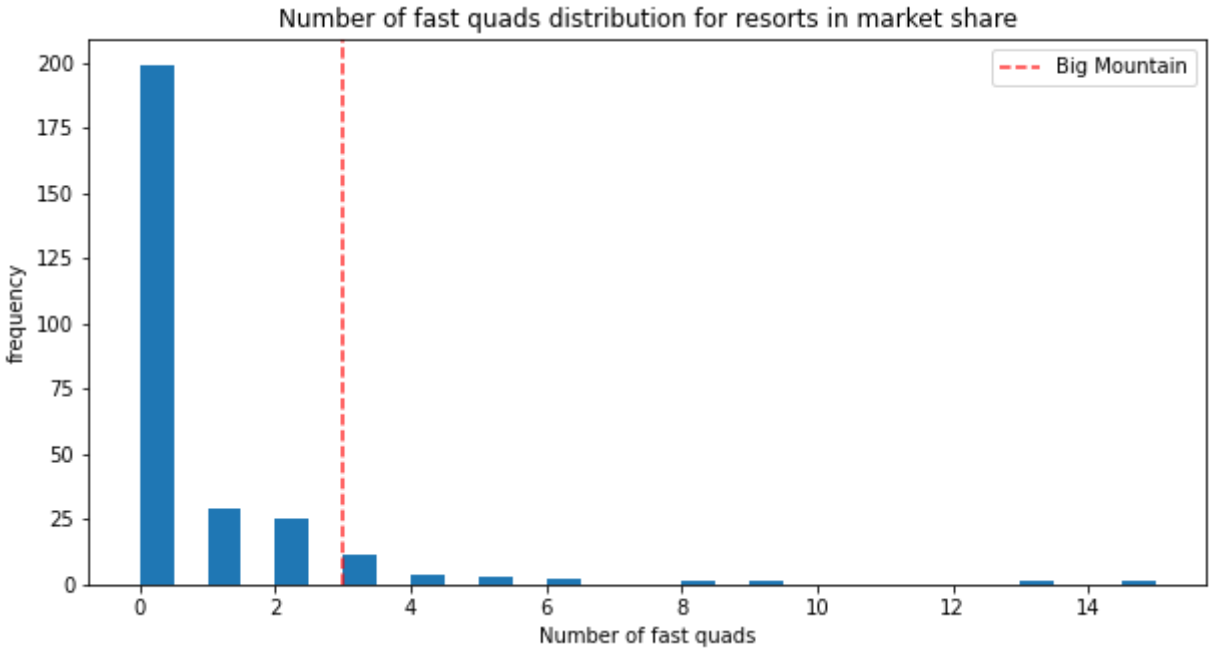
In [21]:
```
plot_compare('total_chairs', 'Total number of chairs')
```



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

5.8.5 Fast quads

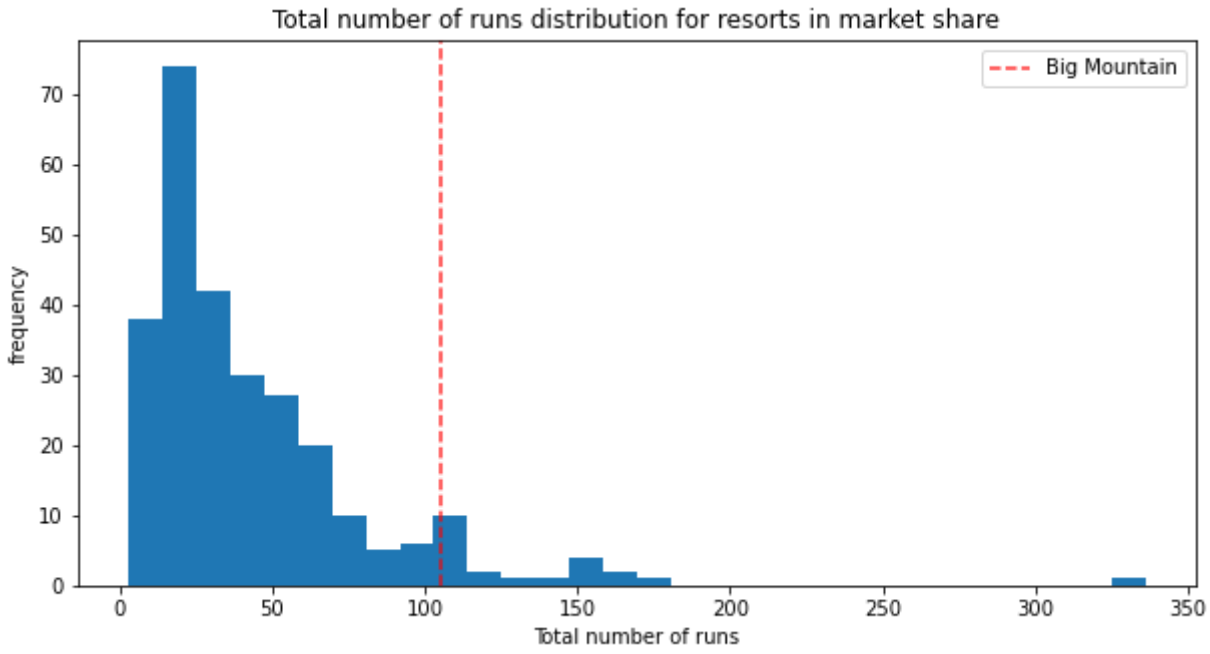
In [22]:
```
plot_compare('fastQuads', 'Number of fast quads')
```



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

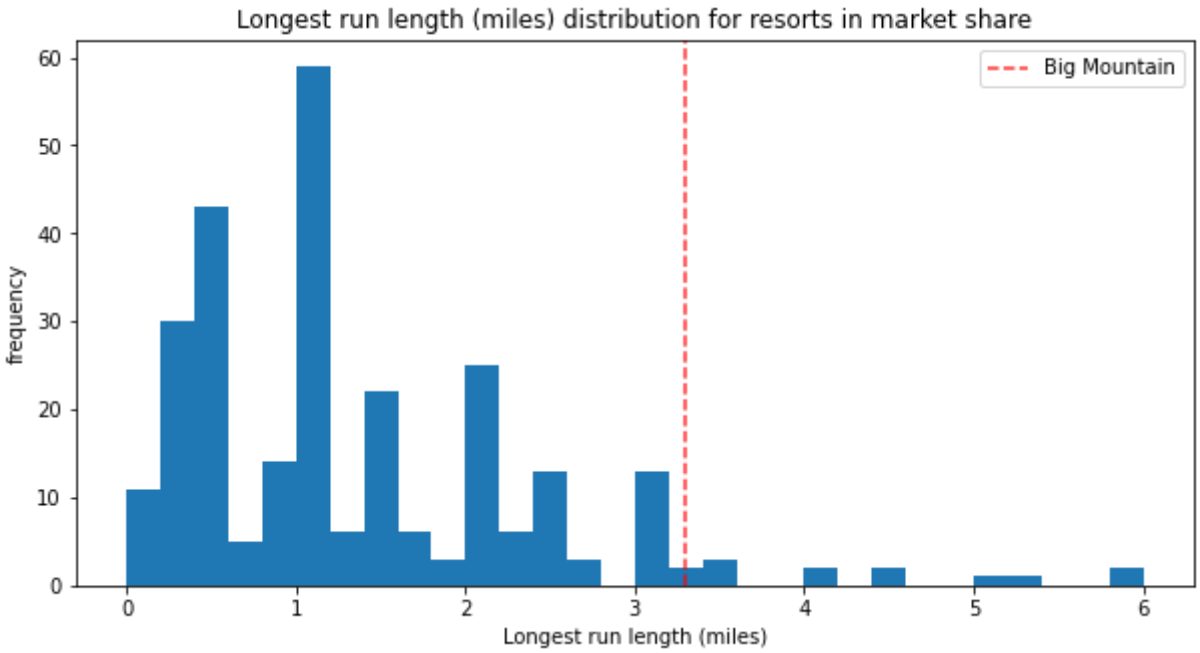
In [23]:
```
plot_compare('Runs', 'Total number of runs')
```



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

5.8.7 Longest run

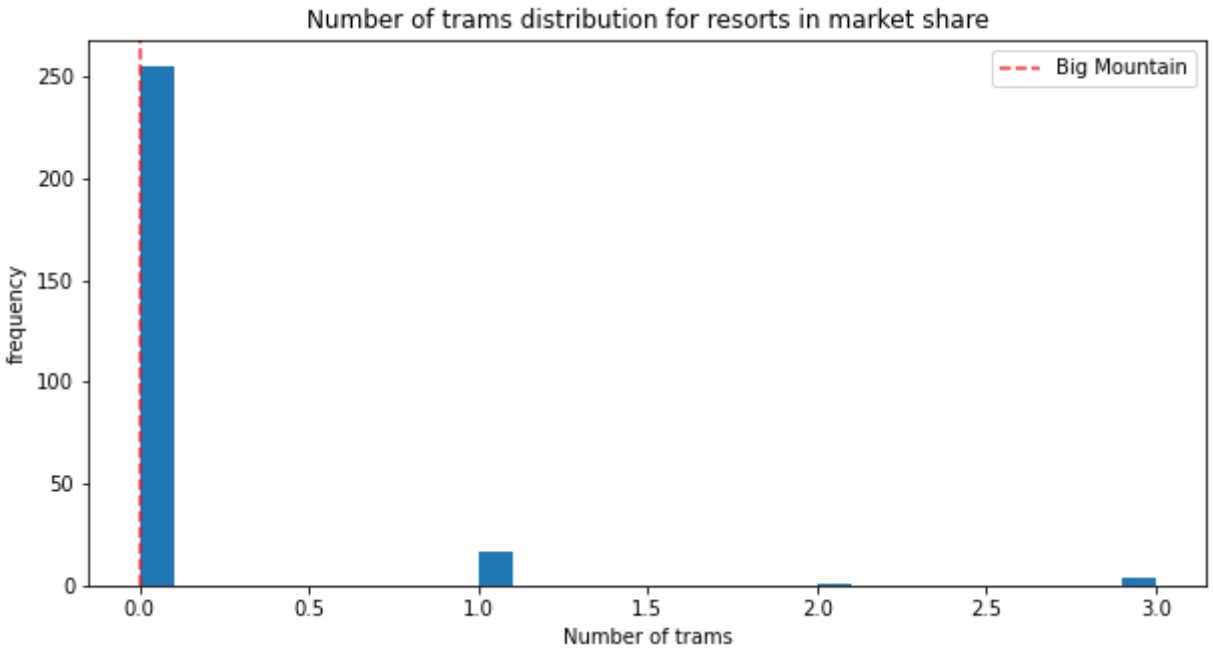
In [24]:
```
plot_compare('LongestRun_mi', 'Longest run length (miles)')
```



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

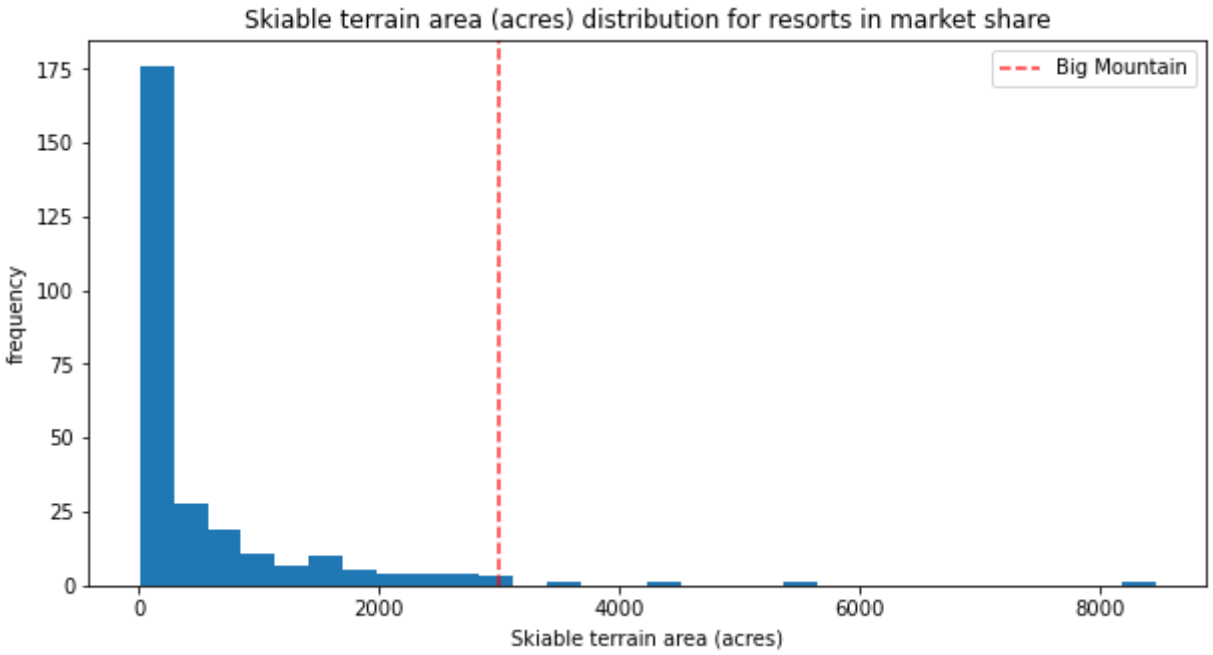
In [25]:
```
plot_compare('trams', 'Number of trams')
```



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

5.8.9 Skiable terrain area

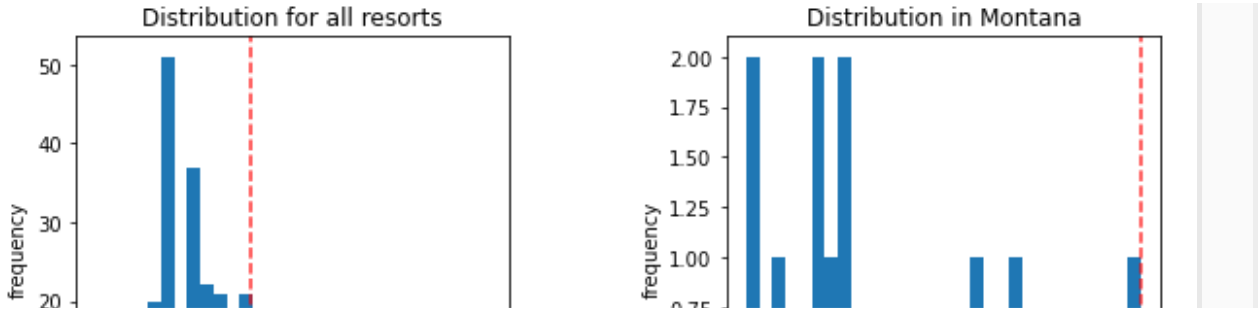
In [26]:
```
plot_compare('SkiableTerrain_ac', 'Skiable terrain area (acres)')
```



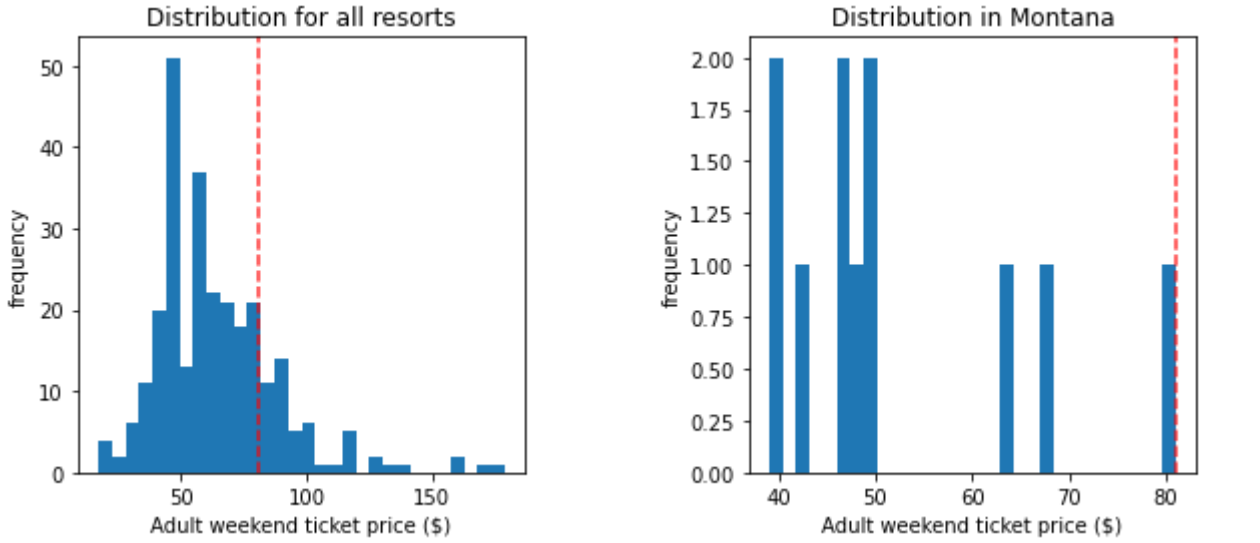
Big Mountain is amongst the resorts with the largest amount of skiable terrain.

Extra exploration: lets look at all these features just in Montana - just to get some perspective

In [29]:
```
# function for plotting features for all resorts and just those in Montana,
def plot_compare_state(feat_name, description, state=None, figsize=(10,4)):
    """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
    """

    fig, ax = plt.subplots(1, 2,figsize=figsize)
    fig.subplots_adjust(wspace=0.5)

    ski_x = ski_data[feat_name]


    # quirk that hist sometimes objects to NaNs, sometimes doesn't
    # filtering only for finite values tidies this up
    ski_x = ski_x[np.isfinite(ski_x)]
    ax[0].hist(ski_x, bins=30)
    ax[0].axvline(x=big_mountain[feat_name].values, c='r', ls='--', alpha=0
    ax[0].set(xlabel=description, ylabel='frequency',  title="Distribution


    ski_x = ski_data.loc[ski_data.state == state, feat_name]
    ski_x = ski_x[np.isfinite(ski_x)]
    ax[1].hist(ski_x, bins=30)
    ax[1].axvline(x=big_mountain[feat_name].values, c='r', ls='--', alpha=0
    ax[1].set(xlabel=description, ylabel='frequency', title='Distribution i
    fig.title = description

plot_compare_state('AdultWeekend', 'Adult weekend ticket price ($)','Montan
plot_compare_state('vertical_drop', 'Vertical drop (feet)','Montana')
plot_compare_state('Snow Making_ac', 'Area covered by snow makers (acres)',
plot_compare_state('total_chairs', 'Total number of chairs','Montana')
plot_compare_state('fastQuads', 'Number of fast quads','Montana')
plot_compare_state('Runs', 'Total number of runs','Montana')
plot_compare_state('LongestRun_mi', 'Longest run length (miles)','Montana')
plot_compare_state('trams', 'Number of trams','Montana')
plot_compare_state('SkiableTerrain_ac', 'Skiable terrain area (acres)','Mon
```



In [30]:
```
plot_compare_state('AdultWeekend', 'Adult weekend ticket price ($)', state=
```



In [83]:
```
# ski_data.AdultWeekend.describe()
# ski_data.AdultWeekend.quantile(.81)
# ski_data.vertical_drop[ski_data.state=="Montana"][ski_data.vertical_drop
print(ski_data.vertical_drop.quantile(.90))
ski_data.vertical_drop[ski_data.Name=="Big Mountain Resort"]
```

```
2363.8
```

Out[83]:
```
124    2353
Name: vertical_drop, dtype: int64
```

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 [34]:
```
expected_visitors = 350000
```

In [35]:
```
all_feats = ['vertical_drop', 'Snow Making_ac', 'total_chairs', 'fastQuads',
             'Runs', 'LongestRun_mi', 'trams', 'SkiableTerrain_ac']
big_mountain[all_feats]
```

Out[35]:

	vertical_drop	Snow Making_ac	total_chairs	fastQuads	Runs	LongestRun_mi	trams	SkiableTerrain
124	2353	600.0	14	3	105.0	3.3	0	300

In [36]:
```
# look at Montana resorts with large vertical drops
f2 = ['Name','state']
f2.extend(all_feats)
ski_data[f2][(ski_data.vertical_drop >2300) & (ski_data.state=='Montana')]
```

Out[36]:

	Name	state	vertical_drop	Snow Making_ac	total_chairs	fastQuads	Runs	LongestRun_mi
115	Bridger Bowl	Montana	2600	100.0	11	0	105.0	1.5
116	Discovery Ski Area	Montana	2380	25.0	8	0	74.0	1.5
120	Montana Snowbowl	Montana	2600	20.0	4	0	37.0	1.2
121	Red Lodge Mountain	Montana	2400	496.0	7	2	70.0	2.5
124	Big Mountain Resort	Montana	2353	600.0	14	3	105.0	3.3

In [37]:
```
bm_test = ski_data.loc[ski_data.Name == "Big Mountain Resort", model.X_colu
bm_test["Runs"]
```

Out[37]:
```
124    105.0
Name: Runs, dtype: float64
```

In [38]:
```
# # NB: Understanding the function
# bm3 = X_bm.copy()
# f1=['Runs']

# # print("before: ",bm_test[f1])
# # bm_test[f1] += 1
# # print("after: ",bm_test[f1])

# # d1=[i for i in range(-1, -4, -1)]
# # print(d1)
# # print(set(zip(f1, d1)))

# for f, d in zip(f1, d1):
#     bm3[f] += d
#     print(f,d)
#     print(bm3[f])
```

In [39]:
```
# NB: Understanding the function
def test_function(features, deltas):
    x=0
    for f, d in zip(features, deltas):
        print("feature:",f)
        print("delta: ", d)
        x+=d
        print("x: ", x)
    return x
```

In [84]:
```
# NB: Understanding the function
# ff=all_feats[0:3]
# test_function(ff, (i for i in range(2,5,1)))

# ivals=(i for i in range(2,5,1))
# test_function(ff, ivals)
```

In [41]:
```
# NB: Understanding the function
# [test_function(ff,[i]) for i in range(2,5,1)]
# Notice it returns a list, with the test_function computed for
# f=all_feats[0] -> only, since only a single d is passed,
# and the function is called for ech value of d in range(2,5,1)
```

In [42]:
```
#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[___] += ___
#     return model.predict(bm2).item() - model.predict(X_bm).item()

    bm2 = X_bm.copy()
#     print(bm2['Runs'])
    for f, d in zip(features, deltas):
#         print("f: ",f)
#         print("d: ",d)
        bm2[f] += d
#         print('new val: ',bm2[f])
#         print('diff: ',model.predict(bm2).item() - model.predict(X_bm).it
    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.

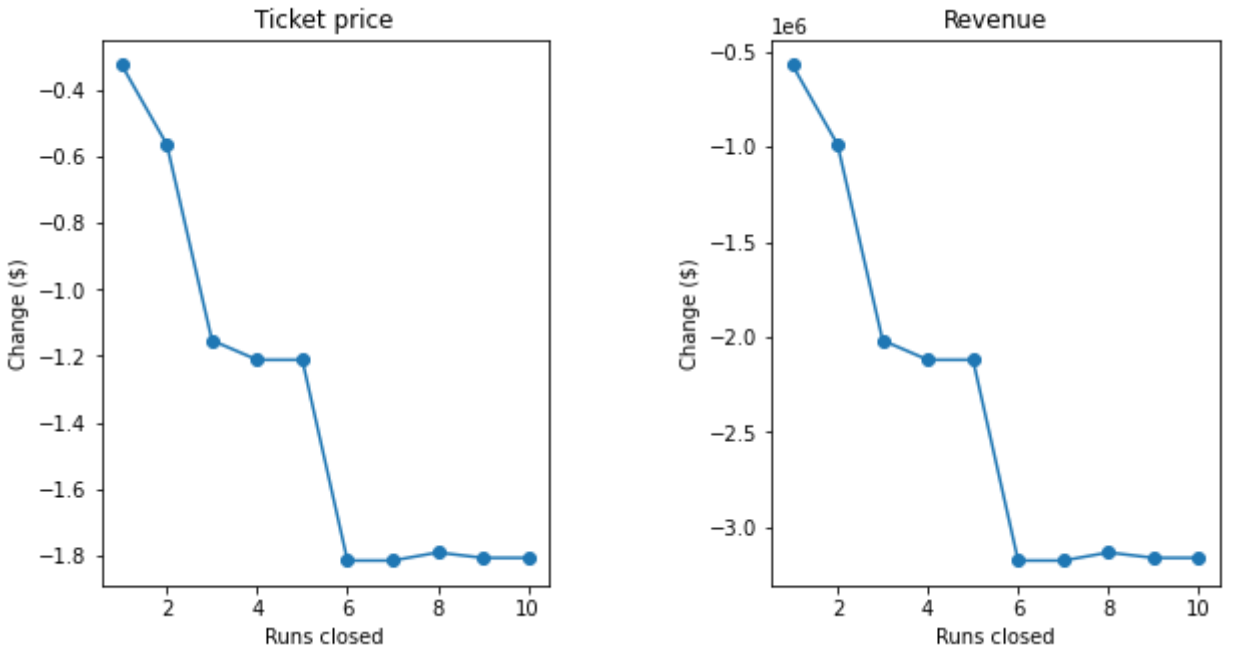
In [43]:
```
# [i for i in range(-1, -11, -1)]
```

In [44]:
```
runs_delta = [i for i in range(-1, -11, -1)]
price_deltas = [predict_increase(['Runs'], [delta]) for delta in runs_delta
```

In [45]:
```
price_deltas
```

Out[45]:
```
[-0.325203252032523,
 -0.5642276422764212,
 -1.1528455284552876,
 -1.2113821138211307,
 -1.2113821138211307,
 -1.8146341463414615,
 -1.8146341463414615,
 -1.7902439024390162,
 -1.8065040650406416,
 -1.8065040650406416]
```

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 * ___ 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');

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.

In [47]:
```
de task 4#
ll `predict_increase` with a list of the features 'Runs', 'vertical_drop',
d associated deltas of 1, 150, and 1
ket2_increase = predict_increase(['Runs', 'vertical_drop', 'total_chairs'],
enue2_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 $0.80
Over the season, this could be expected to amount to $1402846
```

5.9.3 Scenario 3

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

In [49]:
```
sk 5#
scenario 2 conditions, but add an increase of 2 to `Snow Making_ac`
increase = predict_increase(['Runs', 'vertical_drop', 'total_chairs', 'Snow
_increase = 5 * expected_visitors * ticket3_increase
```

In [50]:
```
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 $0.80
Over the season, this could be expected to amount to $1402846
```

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 [51]:
```
#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])
```

Out[51]: 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 ***How much does Big Mountain currently charge?*** Big Mountain currently is charging $81 for an adult weekend ticket. This is the 80th percentile over all resorts in the study, and the highest ticket price charged in the state of Montana.

What does your modelling suggest for a ticket price that could be supported in the marketplace by Big Mountain's facilities? Our model based on the current pricing of resorts across the country prices the current offerings of the Big Mountain Resort at $92.77 for an adult weekend ticket.

This represents an increase in revenue of $20,597,500

How would you approach suggesting such a change to the business leadership?
The modeled price for Big Mountain Resort is $92.77. Assuming the ticket sales remain unchanged, this would indicate an estimated increase in revenue of $20 597 500. Analysis shows that Big Mountain Resort current offerings are well represented in terms of the identified features for which customers are willing to pay a high premium. While several resorts have runs with greater vertical drops, it has the largest skiable terrain, most snow-making area, largest number of chairs, largest number of high-speed quads and the longest ski run in the state of Montana, and these features compare well with the other resorts in general. However, since the ticket prices are already the highest in the state, we recommend testing a more conservative increase of $5 to avoid sudden sticker shock.

Discuss the additional operating cost of the new chair lift per ticket.
Big Mountain estimates the operational cost of a new chair lift to be $1,540,000 per annum. Based on current ticket sales, this cost will be offset by an increase in ticket price of $0.88.

Future improvements

The business has shortlisted some options for either cutting costs or increasing ticket prices:

Scenario 1: Permanently closing down up to 10 of the least used runs. This doesn't impact any other resort statistics.

The model says 1 run may be shut down without any expected change in revenue, particularly if it has little traffic in general. 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 impact to the projeced ticket price. Increasing the closures down to 6 or more leads to significant decrease in the recommended ticket price. These decisions will need further exploration based on the reduction in operating costs from each run closure.

Scenario 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

Big Mountain has a maximum vertical drop of 2350 feet. In Montana, there are 6 resorts that offer a higher vertical drop, maxing at 2600 feet. We can extend the longest run to a point 150 feet lower down, which will require the addition of an extra chair. Our model predicts that this boost in the vertical drop warrants an increase in ticket price of $0.80.

Based on current ticket sales, the estimated $1,540,000 required to cover the annual operating cost of an added chair lift requires an increase in ticket price of $0.88.

Therefore, in order to bear the increase in the operational cost, it will be necessary to leverage the room to increase ticket prices based on Big Mountains other premium offerings. The recommended ticket price of $86 will comfortably compensate for the added annual cost of the additional vertical drop .

Note that we have not considered the corresponding increase in the run length in our estimate.

Scenario 3: Same as number 2, but adding 2 acres of snow making cover

Our analysis shows that the cost of the recommended increase in the snow-making area by 2 acres to support the extended run cannot be offset by a further increase in the ticket price.

Scenario 4: Increase the longest run by 0.2 mile to boast 3.5 miles length, requiring an additional snow making coverage of 4 acres Our model indicates that we cannot support an increase in ticket price by extending the longest run to 3.5 miles, and recommend continuing without extending any run lengths.

5.11 Further work

Q: 2 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?

A: 2 There are a few deficiencies in our analysis. Primarily, we are basing this analysis purely on

existing ticket prices, and have not considered the effect of price on the popularity of a resort. Is the current price, which is much lower than that recommended in the model, offset by increased demand?

Seondly, it would be prudent to try to understand the historical considerations made by the business in setting its existing ticket prices. The marketing or finance team will likely be able to provide some insight. In our analysis, the state was found to be unimportant on a macro level for the purpose of modeling based on existing prices, but does this capture the internal market dynamics? As mentioned before, while the resort has features that make it fairly high-end, the current ticket price is the currently the highest in the state, and this may create a limit to price increases. This will shape the tenor of the presentation to the business, and determine whether to be conservative in the recommendations.

Finally, are the operating costs similar across the resorts? There are other factors beyond the offerings on the slopes, such as accessibility to the resort and the surrounding ammenities, which can affect customer demand and therefore pricing.

Assuming the business leaders felt this model was useful, how would the business make use of it? We might consider providing the business with automated tools that we can create based on our model.

1. A tool that gives the predicted price and confidence bands given the input of changes in the key features of the model.
2. A tool to visually present the distribution of the key features and existing ticket prices across the resorts.