# Big Mountain Resort

## Price Strategy

// Slidesgo Snow

# Table of contents

# Problem Identifications

01

# Problem Statement Worksheet (Hypothesis Formation)

**Provide the optimal pricing strategy which is based on data of 330 resorts in the US and suggest operating changes to increase revenue by ~10% and decrease operating cost by 1-3%.**

## 1 Context

**Big Mountain is a ski resort in Montana. Every year about 350 000 people ski or snowboard at Big Mountain.**
**Big Mountain Resort has recently installed an additional chair lift, which increases operating costs by $ 1 540 000 this season, to help increase the distribution of visitors across mountain.**
**The resort's pricing strategy has been to charge premium above the average price of resorts in its market segments. There is a suspicion that Big Mountain is not capitalizing on its facilities as much as it could.**

## 2 Criteria for success

**The business increases the revenue by 10% and decreases the operating costs by 2%.**

## 3 Scope of solution space

**Consider resorts in the same regions (Subset DataFrame). Find if there are strong correlations between the prices and the base elevations, summits, and vertical drops. If there are strong correlations then project your price based on these correlations. Find out if there strong dependence between fast lifts and prices. Depending on that you can increase the speed of the lifts**

## 4 Constraints within solution space

**If you increase your price too much you can lose loyal customers. If you will be too modest, you will lose possible revenue.**

## 5 Stakeholders to provide key insight

**Jimmy Blackburn – the Director of Operations**
**Alesha Eisen – the Database Manager**

## 6 Key data sources

**A SQL database of 330 resorts in the US**

# Modeling Parameters

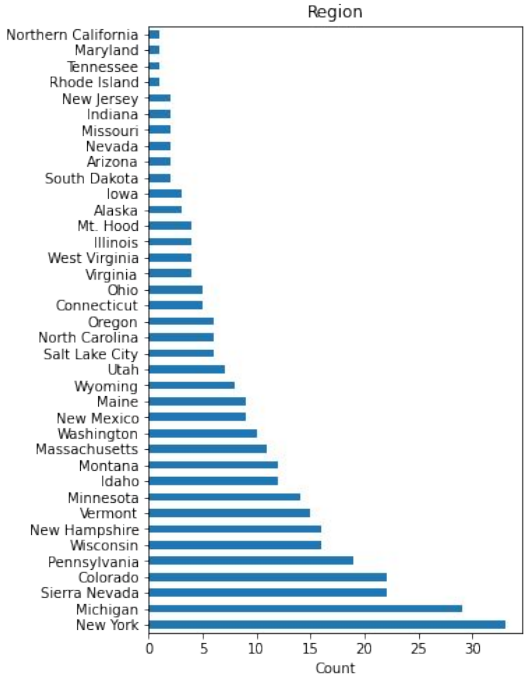
- 3 Categorical Variables
  - Names
  - States
  - Regions (basically same as states)
- 2 Target Variable
  - AdultWeekEnd
  - AdultWeekDay
- 22 Numerical Variable

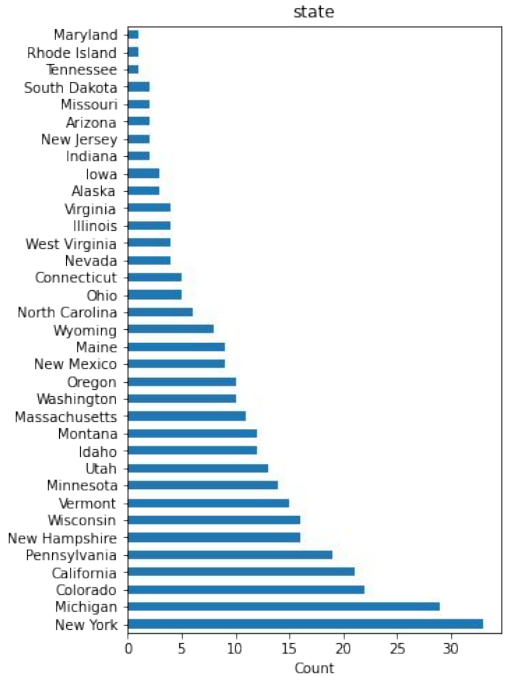
# 02 Data Wrangling

# Categorical Variables

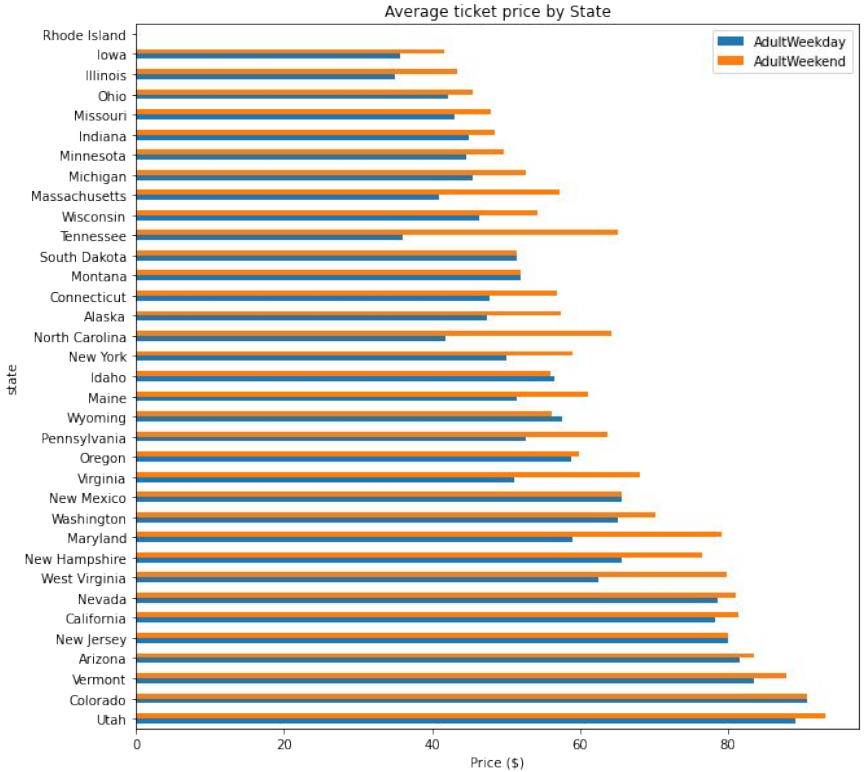
**Combination of Name and State can be used as index for Data, corresponding for unique observation in Data.**

**Let consider distribution along states.**



# Categorical Variables




**Lets also consider the impact of State label to the target variables.**

**Ways to consider using the State information in your modelling include:**

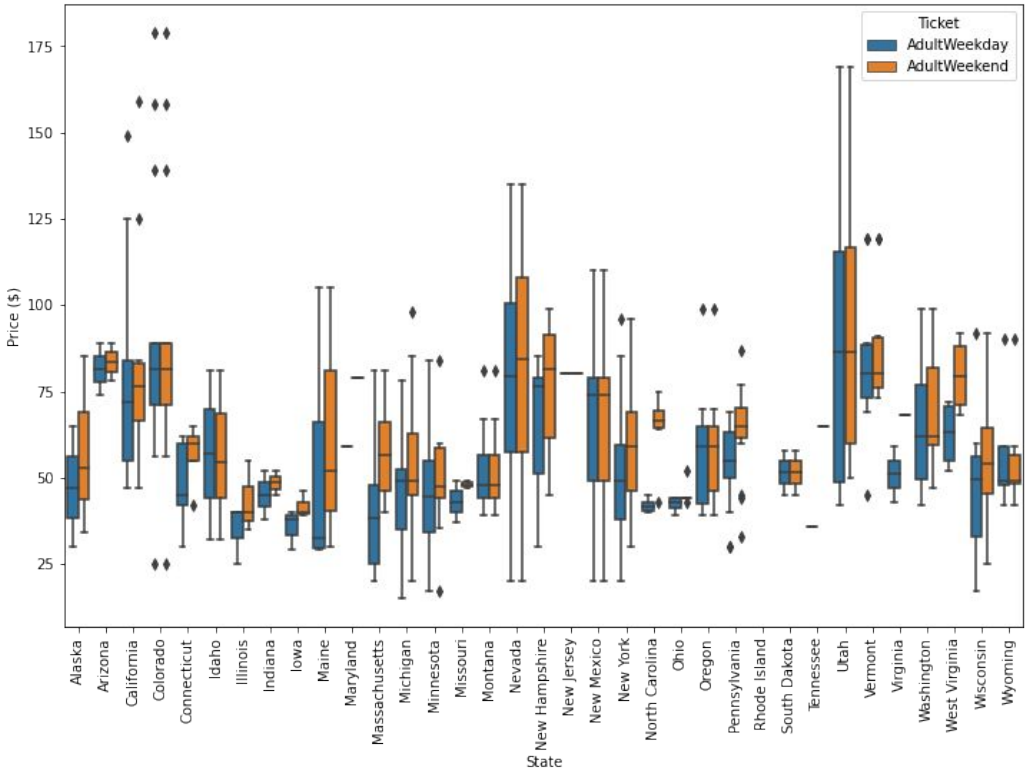
- **disregard State completely**
- **retain all State information**
- **retain State in the form of Montana vs not Montana, as our target resort is in Montana**

# Categorical Variables

It may make sense to allow a model to take into account not just State but also weekend vs weekday.

Thus we currently have two main questions you want to resolve:

- What do you do about the two types of ticket price?
- What do you do about the state information?



# Categorical Variables


**Features that you may be interested in for state-wide summary are:**

- **TerrainParks**
- **SkiableTerrain_ac**
- **daysOpenLastYear**
- **NightSkiing_ac**

**Calculate some state-wide summary statistics for later use.**

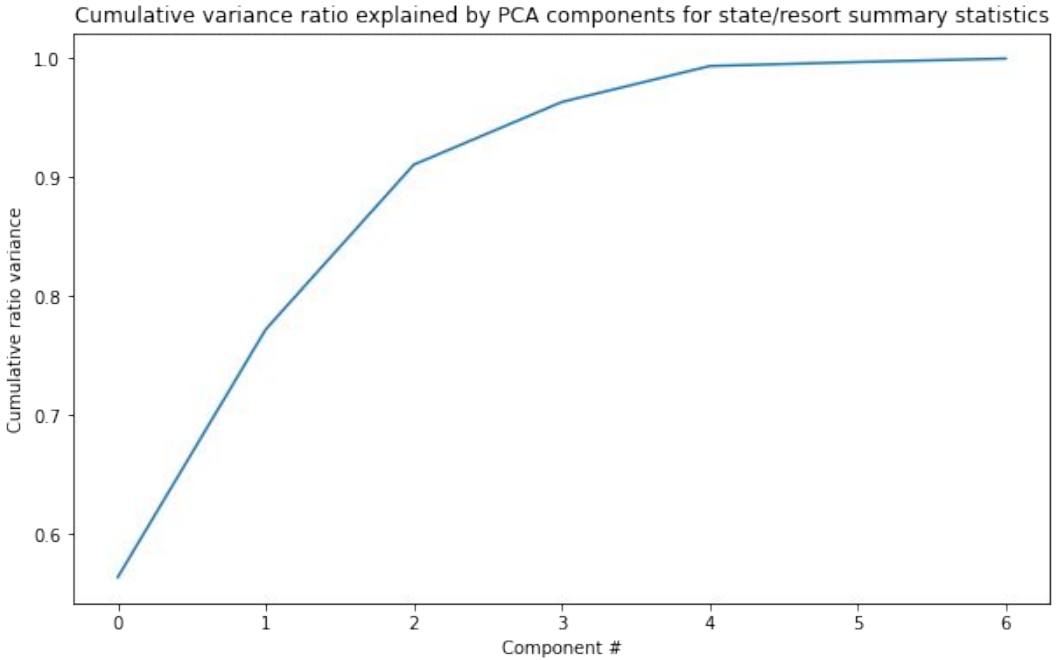
**Population and area data for the US states can be obtained from wikipedia.**

**You could be interested in the ratio of resorts serving a given population or a given area.**

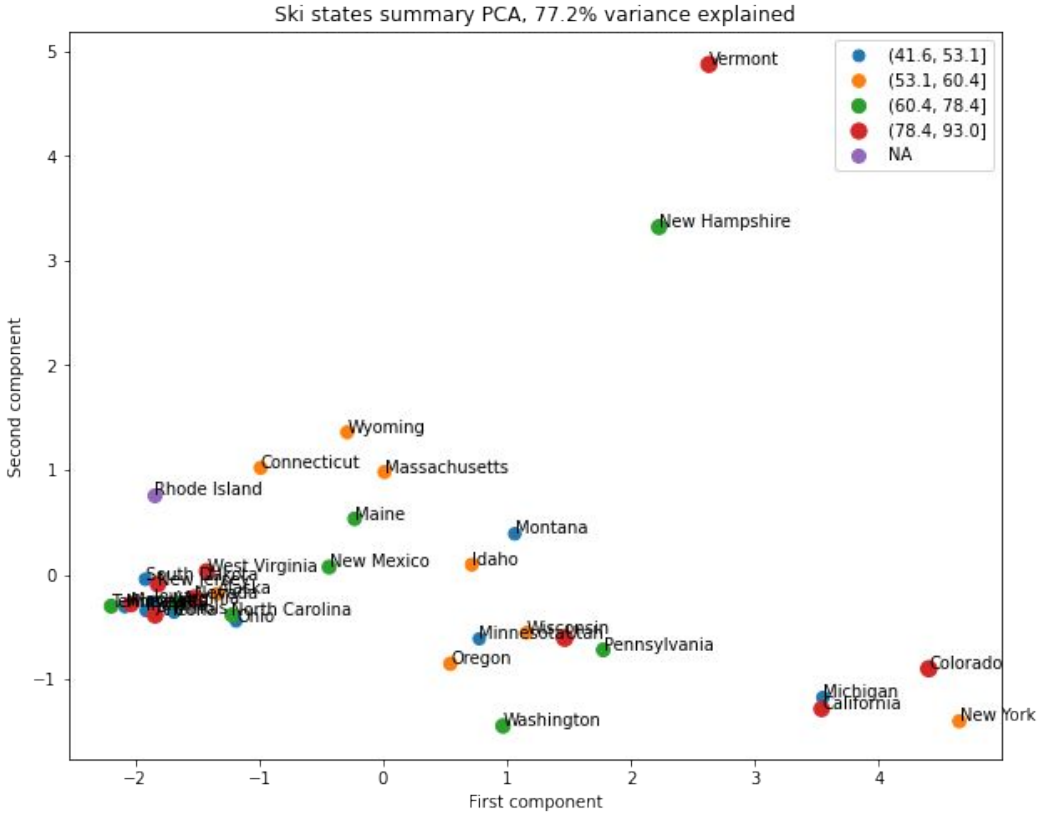
# Categorical Variables

**One way to disentangle interconnected web of relationships is via principle components analysis (PCA). The basic steps in this process are:**

1. **scale the data (important here because our features are heterogenous)**
2. **fit the PCA transformation (learn the transformation from the data)**
3. **apply the transformation to the data to create the derived features**
4. **(optionally) use the derived features to look for patterns in the data and explore the coefficients**



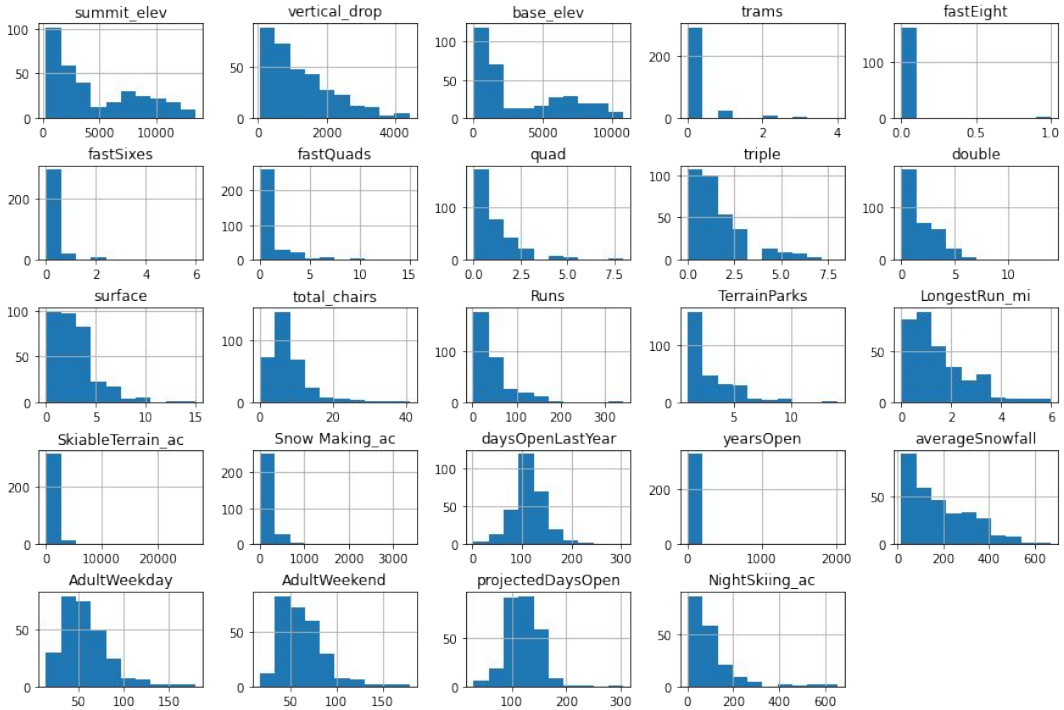
# Categorical Variables




**In this representation of the ski summaries for each state, which accounts for some 77% of the variance, you simply do not seeing a pattern with price.**

**Therefore we are treating all states equally, and work towards building a pricing model that considers all states together, without treating any one particularly specially.**

# Numerical Features

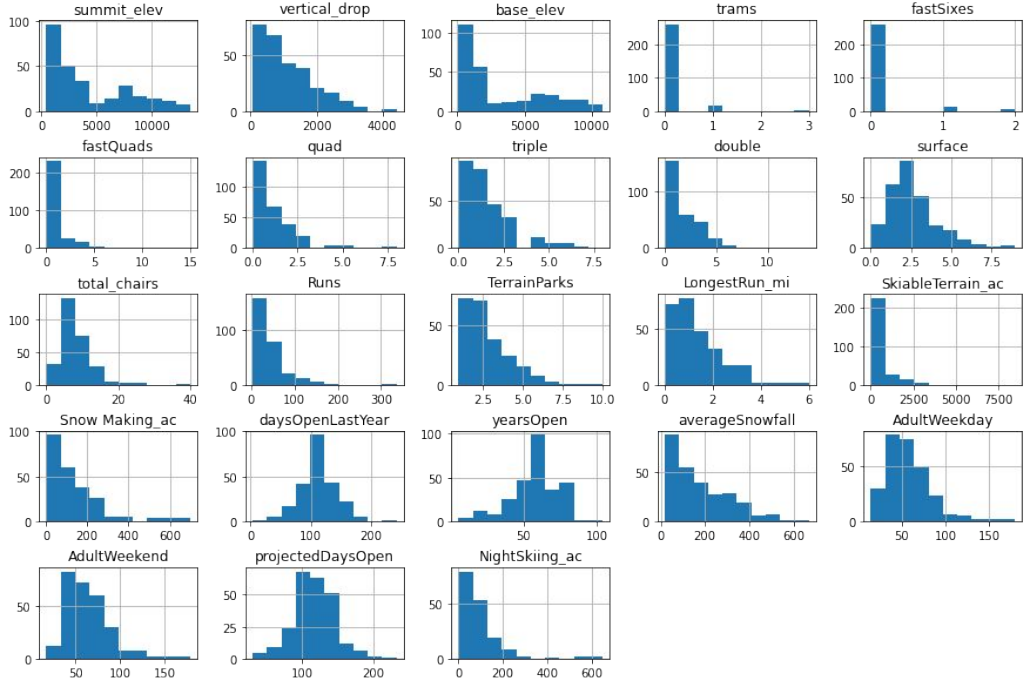



**What features do we have possible cause for concern about and why?**

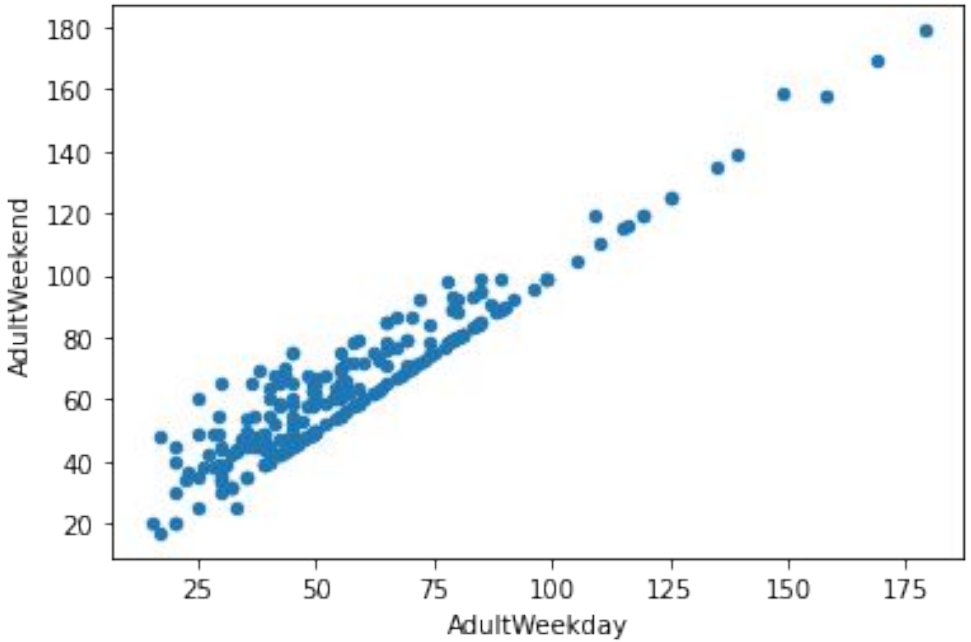
- **SkiableTerrain_ac because values are clustered down the low end;**
- **Snow Making_ac for the same reason;**
- **fastEight because all but one value is 0 so it has very little variance, and half the values are missing;**
- **fastSixes raises an amber flag; it has more variability, but still mostly 0;**
- **trams also may get an amber flag for the same reason;**
- **yearsOpen because most values are low but it has a maximum of 2019, which strongly suggests someone recorded calendar year rather than number of years.**

# Numerical Features

- Recheck outliers (namely "silverton mountain skiable area"). Suspicious big value 26819, but the checked value is 1819. Correct data;
- You have no ticket pricing information at all for this resort. You'll simply be dropping the entire row!
- There is essentially no information in this column. Drop it;
- Not investigate further;
- Not investigate further;
- You can certainly state that no resort will have been open for 2019 years! it feels best to simply drop this row.

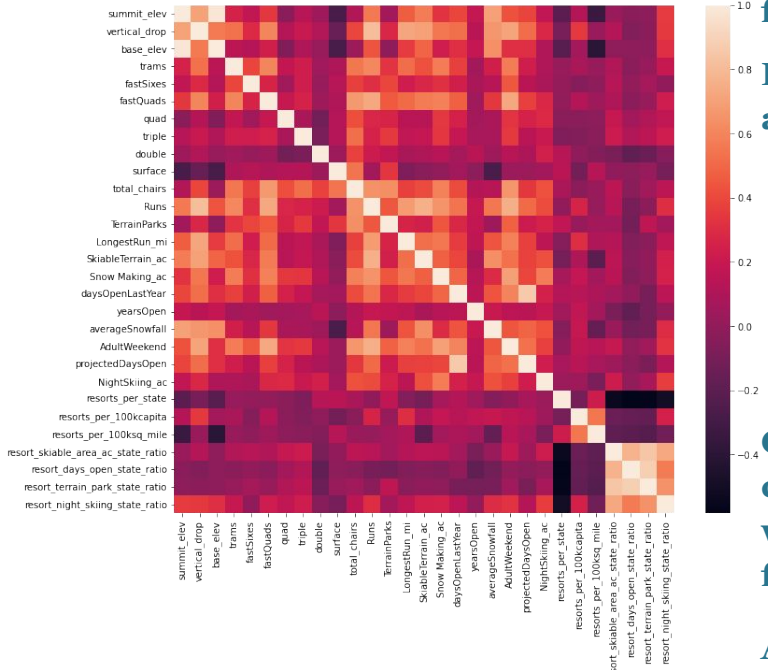


# Target Variables




**Weekend prices have the least missing values of the two, so drop the weekday prices and then keep just the rows that have weekend price.**

# Numerical Features




It's now time to merge the two datasets and engineer some intuitive features.

Having merged your state summary features into the ski resort data, add "state resort competition" features:

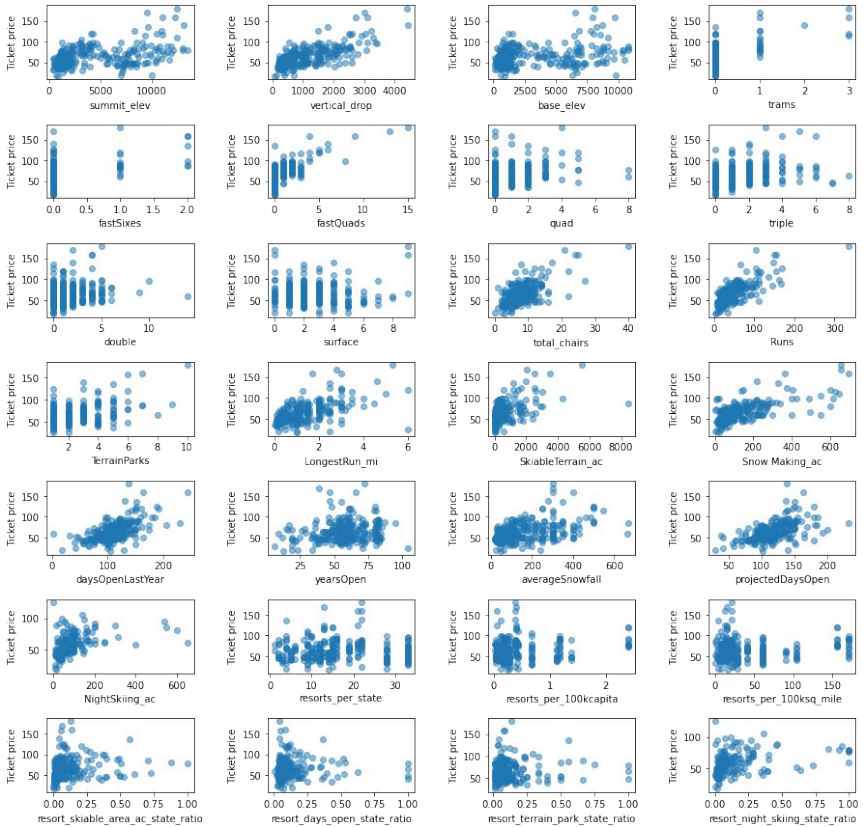
- ratio of resort skiable area to total state skiable area
- ratio of resort days open to total state days open
- ratio of resort terrain park count to total state terrain park count
- ratio of resort night skiing area to total state night skiing area

Once you've derived these features to put each resort within the context of its state,drop those state columns. Their main purpose was to understand what share of states' skiing "assets" is accounted for by each resort.

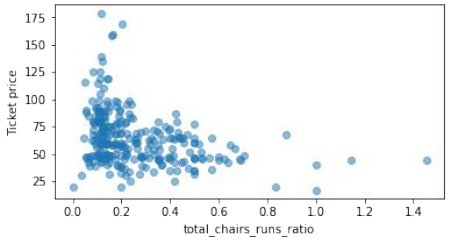
A great way to gain a high level view of relationships amongst the features.

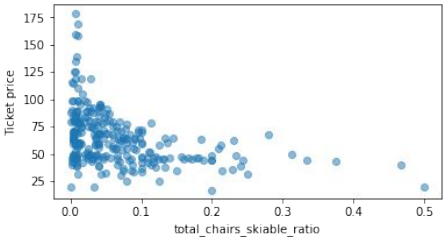
# Numerical Features

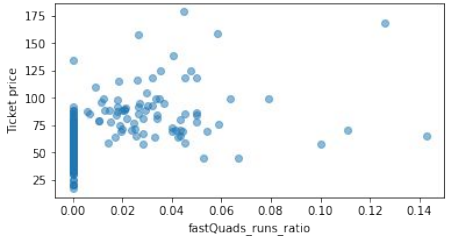
In the scatterplots you see what some of the high correlations were clearly picking up on. There's a strong positive correlation with vertical_drop. fastQuads seems very useful. Runs and total_chairs appear quite similar and also useful. resorts_per_100kcapita shows something interesting that you don't see from just a headline correlation figure. When the value is low, there is quite a variability in ticket price, although it's capable of going quite high. Ticket price may drop a little before then climbing upwards as the number of resorts per capita increases. Ticket price could climb with the number of resorts serving a population because it indicates a popular area for skiing with plenty of demand. The lower ticket price when fewer resorts serve a population may similarly be because it's a less popular state for skiing. The high price for some resorts when resorts are rare (relative to the population size) may indicate areas where a small number of resorts can benefit from a monopoly effect.

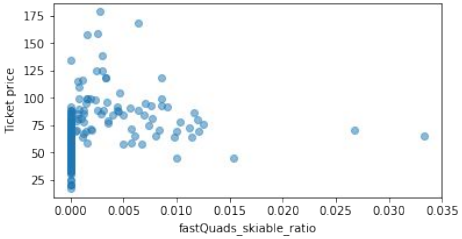


# Numerical Features




**At first these relationships are quite counterintuitive. It seems that the more chairs a resort has to move people around, relative to the number of runs, ticket price rapidly plummets and stays low. What we may be seeing here is an exclusive vs. mass market resort effect; if you don't have so many chairs, you can charge more for your tickets, although with fewer chairs you're inevitably going to be able to serve fewer visitors. Your price per visitor is high but your number of visitors may be low. Something very useful that's missing from the data is the number of visitors per year.**

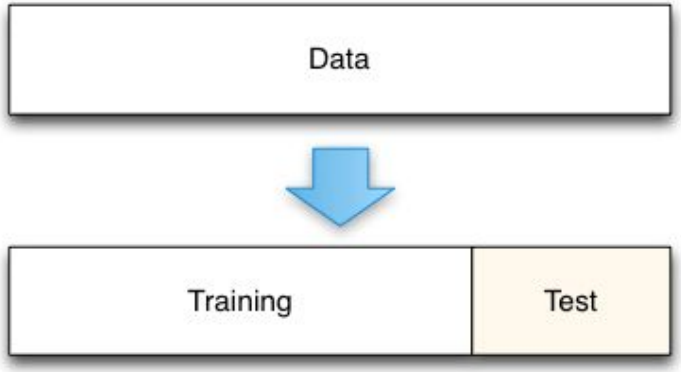
**It also appears that having no fast quads may limit the ticket price, but if your resort covers a wide area then getting a small number of fast quads may be beneficial to ticket price.**

# 03

# Preprocessing and Training

# Train/Test Split

**In machine learning, when you train your model on all of your data, you end up with no data set aside to evaluate model performance. You could keep making more and more complex models that fit the data better and better and not realise you were overfitting to that one set of samples. By partitioning the data into training and testing splits, without letting a model (or missing-value imputation) learn anything about the test split, you have a somewhat independent assessment of how your model might perform in the future.**



# Metrics

There are many ways of assessing how good one set of values agrees with another, which brings us to the subject of metrics.

The mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

The total sum of squares (error) $SS_{tot} = \sum_i (y_i - \bar{y})^2$

The residual sum of squares $SS_{res} = \sum_i (y_i - \hat{y})^2$

The coefficient of determination (R2-score) $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

Mean Absolute Error $MAE = \frac{1}{n}\sum_i |y_i - \hat{y}|$

Mean Squared Error $MSE = \frac{1}{n}\sum_i (y_i - \hat{y})^2$

# Imputing Missing Features

**You can impute missing values using scikit-learn, but note that you should learn values to impute from a train split and apply that to the test split to then assess how well your imputation worked.**

- ★ **Your first thought might be to impute missing values using the median or the mean.**
- ★ **As you have features measured in many different units, with numbers that vary by orders of magnitude, start off by scaling them to put them all on a consistent scale. The StandardScaler scales each feature to zero mean and unit variance.**
- ★ **Train the model on the train split.**
- ★ **Assess model performance.**

  **For the Median Imputer**
  - ❖ **R2-score = (0.8177, 0.7209)**
  - ❖ **MAE = (8.5478, 9.4070)**
  - ❖ **MSE = (111.8958, 161.7315)**

  **For the Mean Imputer**
  - ❖ **R2-score = (0.8170, 0.7163)**
  - ❖ **MAE = (8.5368, 9.4163)**
  - ❖ **MSE = (112.3769, 164.3926)**

# Best Features Selection

You suspected the model was overfitting. This is no real surprise given the number of features you blindly used. It's likely a judicious subset of features would generalize better. `sklearn` has a number of feature selection functions available. The one you'll use here is `SelectKBest` which, as you might guess, selects the k best features.

Clearly selecting a subset of features has an impact on performance. `SelectKBest` defaults to k=10. You've just seen that 10 is worse than using all features. What is the best k?

# Cross Validation

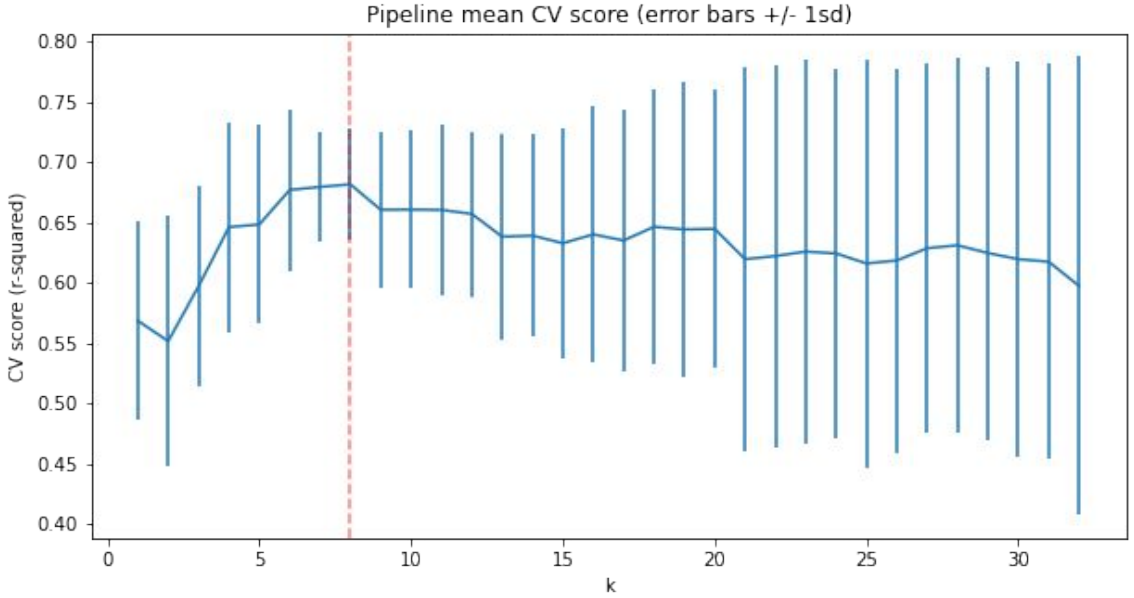
You could keep going, trying different values of k, training a model, measuring performance on the test set, and then picking the model with the best test set performance. There's a fundamental problem with this approach: *you're tuning the model to the arbitrary test set*! If you continue this way you'll end up with a model works well on the particular quirks of our test set *but fails to generalize to new data*. The whole point of keeping a test set is for it to be a set of that new data, to check how well our model might perform on data it hasn't seen.

The way around this is a technique called *cross-validation*. You partition the training set into k folds, train our model on k-1 of those folds, and calculate performance on the fold not used in training. This procedure then cycles through k times with a different fold held back each time. Thus you end up building k models on k sets of data with k estimates of how the model performs on unseen data but without having to touch the test set.

# Hyperparameter Search

Now you have a range of k to investigate. Is 1 feature best? 2? 3? 4? All of them? You could write a for loop and iterate over each possible value, doing all the housekeeping ourselves to track the best value of k. But this is a common task so there's a built in function in sklearn. This is GridSearchCV. This takes the pipeline object, in fact it takes anything with a .fit() and .predict() method. In simple cases with no feature selection or imputation or feature scaling etc. you may see the classifier or regressor object itself directly passed into GridSearchCV. The other key input is the parameters and values to search over. Optional parameters include the cross-validation strategy and number of CPUs to use.

# Hyperparameter Search




**There was an initial rapid increase with k, followed by a slow decline. Also noticeable is the variance of the results greatly increase above k=8. As you increasingly overfit, expect greater swings in performance as different points move in and out of the train/test folds.**

# Final Model Selection

You built a best linear model and a best random forest model. You need to finally choose between them. You can calculate the mean absolute error using cross-validation.

For Linear Regression MAE = 11.7934656686693 27.

For Random Forest MAE = 9.537730050637332.

The random forest model has a lower cross-validation mean absolute error by almost $1.

# Data Quantity Assessment

**Would more data be useful? We're often led to believe more data is always good, but gathering data invariably has a cost associated with it. Assess this trade off by seeing how performance varies with differing data set sizes.**

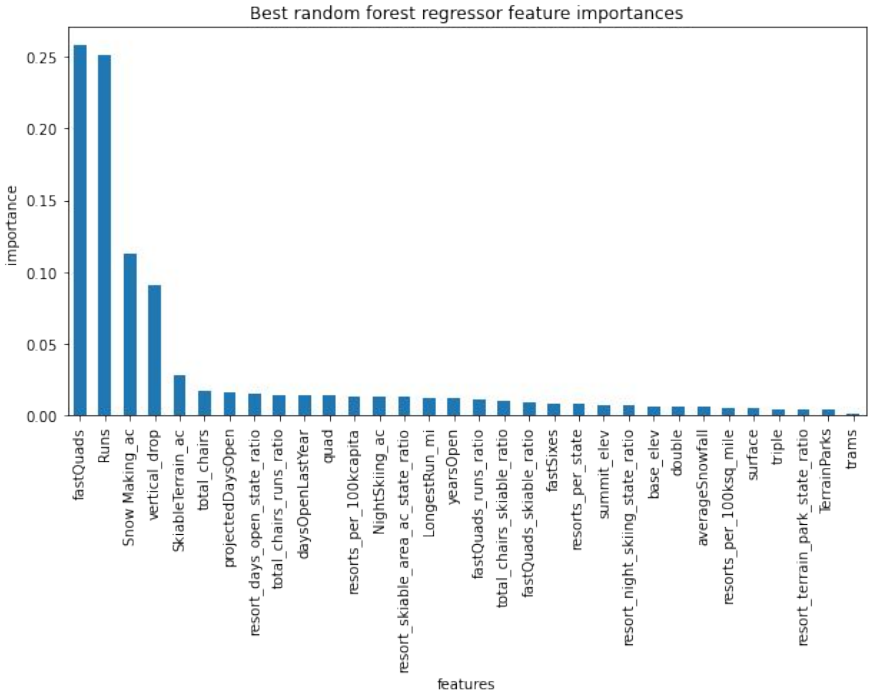
**This shows that you seem to have plenty of data. There's an initial rapid improvement in model scores as one would expect, but it's essentially levelled off by around a sample size of 40-50.**

# Recommendation And Key Findings

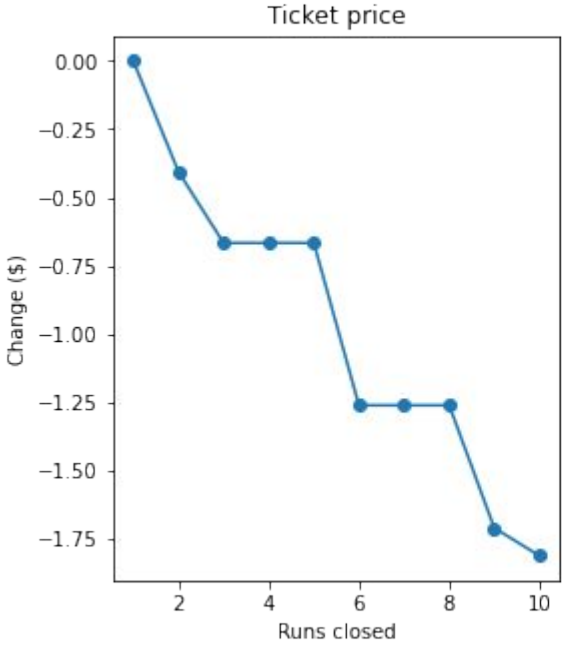
04

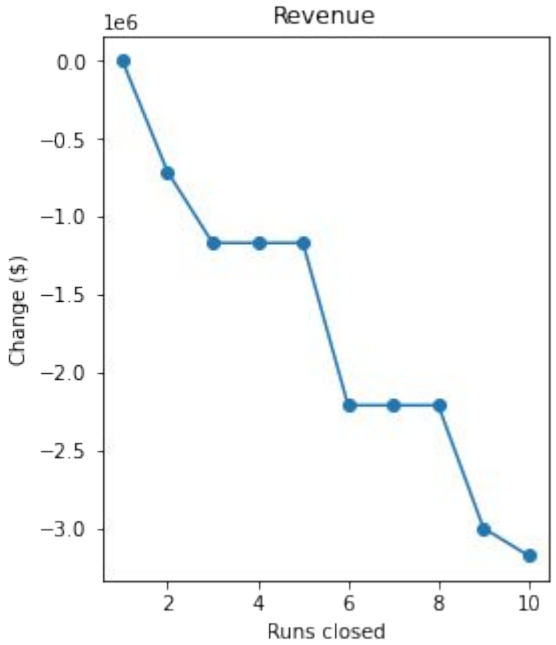
# Focus on

1) The number of four persons chairs;
2) Count of the number of runs on the resorts;
3) Total area covered by snow making machines in acres;
4) Vertical change in elevation from the summit to the base in feet;
5) Sum of all chairlifts at the resorts.



# Close one Run




**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.**

# 05

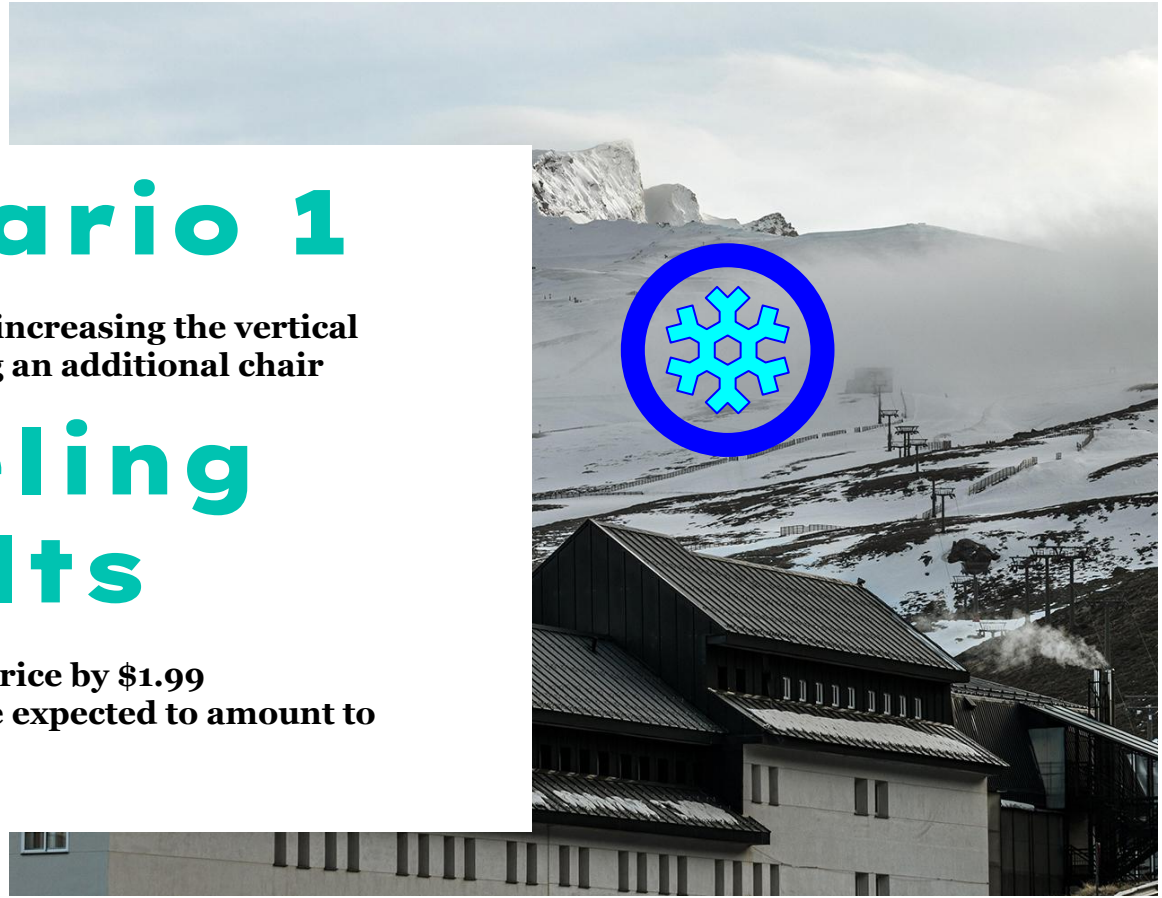
# Modeling Results and Analysis

# Scenario 1

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

# Modeling Results

**Increases support for ticket price by $1.99**
**Over the season, this could be expected to amount to $3474638**



# Scenario 2

**Repeating the previous one but adding 2 acres of snow making.**

# Modeling Results

**Increases support for ticket price by $1.99**
**Over the season, this could be expected to amount to $3474638**

# Scenario 3

**Increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability.**

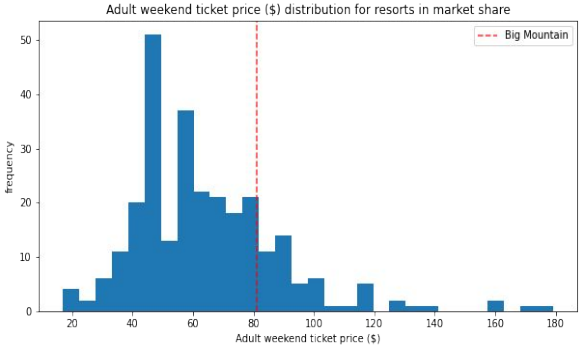
# Modeling Results

**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.**

# Summary and Conclusion

06

- The State labels don’t affect Price Much
- Big Mountain Resort modelled price is $95.87, actual price is $81.00. Even with expected mean absolute error of $10.39, this suggests there is room for increase.




# Thank you for Attention

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**