Big Mountain Resort Montana

Libi Voshin

Springboard

This project report will focus on :

Which pricing strategy should Big Mountain Resort apply in the upcoming season to increase its Revenues and return its investment of $1.54 Million?

GUIDED CAPSTONE PROJECT REPORT – Libi Voshin

Problem identification

Our data science team was brought in to implement a data-driven model that would help to answer the following business questions.

Recently resort invested in a chair-lift, which has increased its operating costs by $1.54M this season. Considering this new addition, to maximize its returns relative to its position in the market, we investigated **What price should Big Mountain Resort charge for their ticket?** Resort's currently pricing strategy has been to charge a premium above the average cost of alternatives in its market segment. This report will present answers to whether it can increase the price even higher and, if so, in how much, so visitors will still be willing to pay the price.

There's a suspicion that Big Mountain is not capitalizing on its facilities as much as it could. We will be helping to determine **How to capitalize on its facilities in the best possible way**. Resort's management has presented us with four changes that they are considering. Our model tested these propositions and predicted how each scenario would influence current consumers' willingness to pay for a ticket. I will present **Which of the considered changes will cut costs without undermining the ticket price and which change will support an even higher ticket price**.

Finally, based on the data we have gathered on the existing list of facilities on 330 resorts belonging to the same market share (See appendix B- assumptions section), we will provide a user-friendly model that would support future business decisions regarding the influence of future change to resort's facilities on ticket value, As well as, **How to adjust their ticket price based on those changes**.

Recommendation and key findings

Since each visitor on average buying 5-day tickets and each year there are about 350,000 visitors at the resort, Our modeling suggests that charging **$94.22** per ticket could be fairly supported in the marketplace by Big Mountain's facilities. Solely to cover the resort's recent investment, it should raise ticket price by $0.88 per ticket.

Features that came up as highly valued by customers (see Appendix A) include:

- Number of Fast Quads
- Number of Runs
- Snow Making area
- Vertical drop
- Skiable terrain area
- Total number of chairs

After seeing where Big mountain resort stands amongst those areas, we feel confident that existing resorts' facilities can support the higher ticket price.

For further improvements, we would recommend the following:

1. First, to permanently close the one least used run, as shown from the model that it is not reducing ticket value.
2. Then, our model shows that increase the vertical drop by 150 feet by adding a lower run supports a $1.99 increase in the ticket price, which over the season could be expected to amount to revenue of $3.475 M. Implementation of this scenario requires the installation of an additional chair-lift to bring skiers back up. Resort's previous experience shows that the price should be raised to $0.88 per ticket to cover additional chair-lift yearly operation costs.

   Hence, It is an **extra profit of more than $2M** (not include the additional chair-lift installation cost).

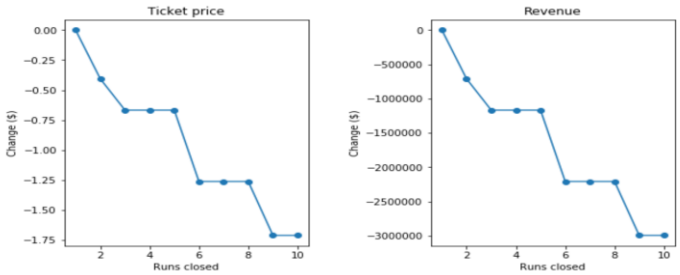
   Note that only after knowing the additional chair-lifts yearly operational costs could we calculate the profit.

It is also worth mentioning that adding 2 acres of snowmaking coverage makes no difference in ticket value. Therefore, it is not worth the investment.

3. Lastly, the resort should look up at its runs' operational costs and make a decision when should it permanently close its least used runs according to this decision table:

Condition: Yearly operational costs of..	**Decision**: Permanently close down..
2 runs > $675K	2 least used runs
5 runs > $ 1.225 M	5 least used runs
8 runs > $ 2.2 M	8 least used runs

As we can see in the graph below, If Big Mountain closes down three runs, it seems they may as well close down 4 or 5 as there's no further loss in the ticket. So as 6,7 and 8.



Summary and conclusion

Currently, Big Mountain charges $81, and although state-wise, Big Mountain's ticket price is the highest, and so it sits high amongst all resorts, there are still resorts with a higher price and up to double the price. Note that this relies on the implicit assumption that all other resorts are primarily setting prices based on how much people value certain facilities. Essentially this assumes a free market sets prices.

As we saw, the 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 some facilities and less for others. Being able to sense how facilities support a given ticket price is valuable business intelligence. Thus, the utility of our model comes in.

For future queries and new scenarios examinations, management can use our model to obtain the predicted increase of ticket prices from each unique scenario. all that is needed is to insert a list of all the features/facilities that will be affected by the new plan and their corresponding deltas, to the function that we supplied ("predict_increase") to receive the predicted increase in $ (or decrease for negative results).

Appendix A

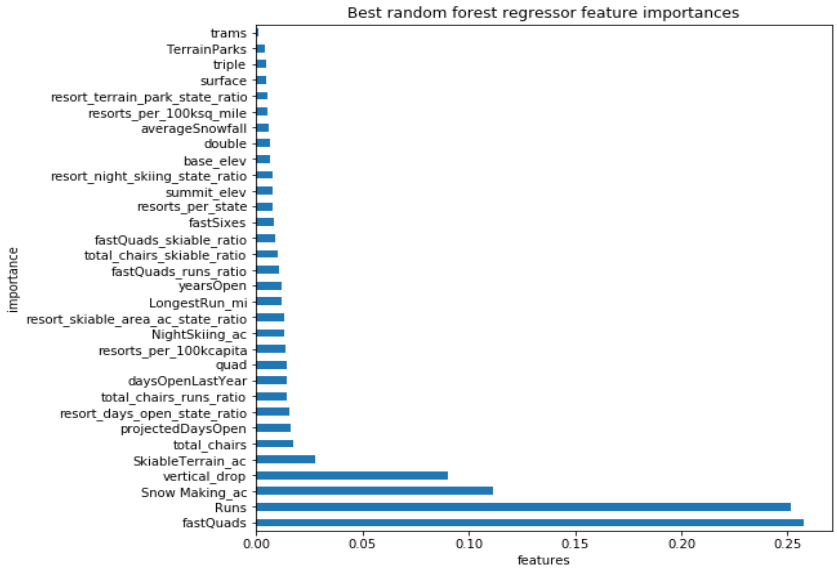
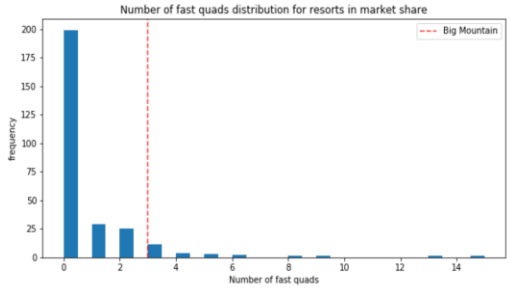
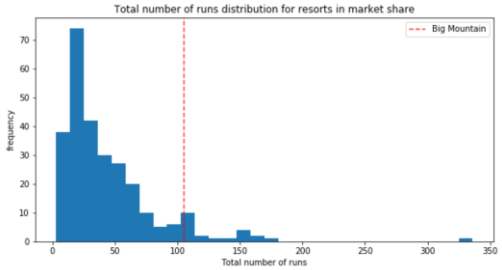


Figure 1. Features that came up as highly valued by customers in our best modeling

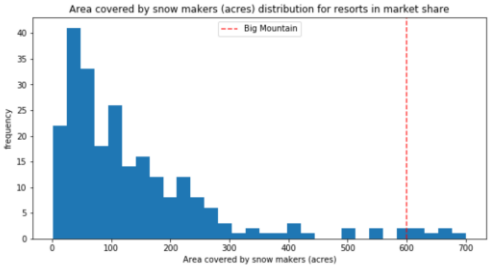
1. The number of Fast Quads - Most resorts have no fast quads. Big Mountain has 3.



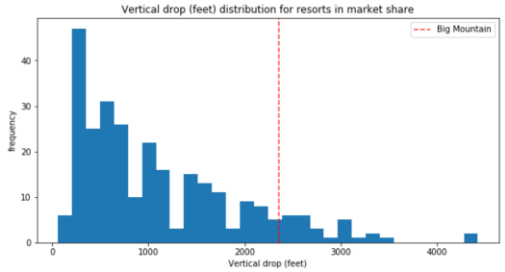
2. Runs - Big Mountain compares well for the number of runs



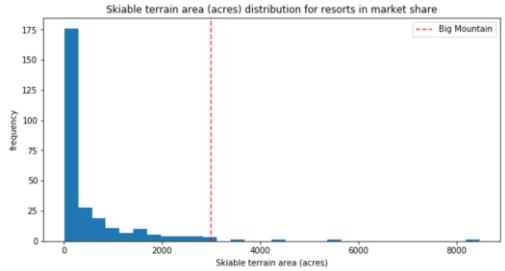
3. Snow Making area – Big Mountain resort is very high up the league table of snowmaking areas.



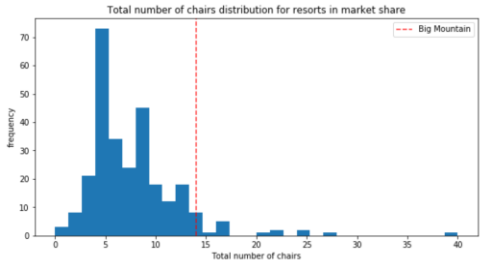
4. Vertical drop - Resort is doing well, and there are only several others with a more significant drop



5. Skiable terrain area - Big Mountain is amongst the resorts with an enormous amount of skiable terrain.



6. Total number of chairs - Big Mountain has amongst the highest total number of chairs



Appendix B

Assumptions

Data quality and "data correction" steps

Data quantity assessment

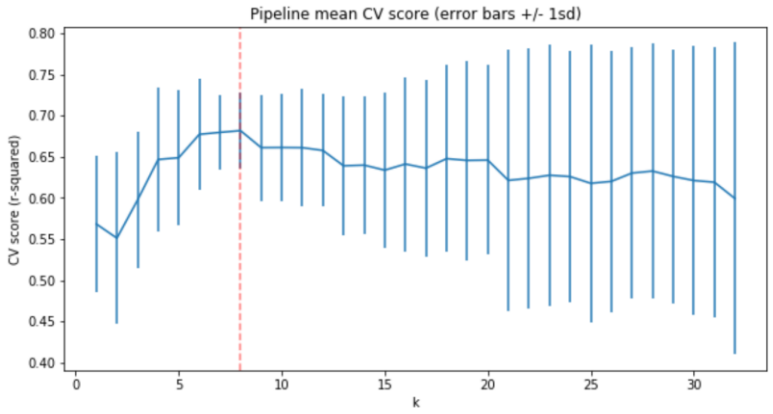
Ski Resort Features we worked on included:

Models' description

- **We built a best linear model and a best random forest model**
- The random forest model has lower cross-validation mean absolute error by almost $1 on the train set and a lower MAE on a test set by more than $2 (Verifying performance on the test set produces performance consistent with the cross-validation results). It also exhibits less variability.

	Model	Imputing missing values technic	Standard scale	Selection of K	R^2 Performance	R^2 Performance on Test set after 5-Fold Cross Validation on Train set		Mean Absolute Error performance after 5-Fold Cross-Validation on Train set	
					Train, Test	mean , std	range of R^2 (mean-/+2std)	Train (mean, std)	Test (mean)
1	Linear Regression	Median	Yes	-	0.818 , 0.721	-	-	8.548	9.407
2	Linear Regression	Mean	Yes	-	0.817 , 0.716	-	-		-
3	Linear Regression	Median	Yes	K=10	0.767 , 0.626	0.6606, 0.0657	[0.53, 0.79]	9.502	11.202
4	Linear Regression	Median	Yes	K=15	0.792 , 0.638	0.6327, 0.0950	[0.44, 0.82]	9.21176	10.488246
5	Linear Regression	Median	Yes	K=8	0.762, 0.597	0.6815 , 0.0459	[0.59, 0.77]	10.499, 1.622	11.7935
6	Random Forest	Median	Yes	default	-	0.6385, 0.1444	[0.3497 , 0.927]		-
7	Random Forest	Median	None	number trees: 69	-	0.7082, 0.0656	[0.5769 , 0.8395]	9.659, 1.349	9.4955

we can also see in the scheme below, there was an initial rapid increase in the score with increase in k, followed by a slow decline after K=8. also noticeable that the variance of the results greatly increases above k=8.



As we are increasingly overfit. we can expect greater swings in performance and that can explain results that seems to be better (model 1 to 4 vs model 5)

Appendix C

GUIDED CAPSTONE PROJECT REPORT – Libi Voshin

Take aways during – Data Wrangling phase

Number of missing values by column

	count	%
fastEight	166	50.303030
NightSkiing_ac	143	43.333333
AdultWeekday	54	16.363636
AdultWeekend	51	15.454545
daysOpenLastYear	51	15.454545
TerrainParks	51	15.454545
projectedDaysOpen	47	14.242424
Snow Making_ac	46	13.939394
averageSnowfall	14	4.242424
LongestRun_mi	5	1.515152
Runs	4	1.212121
SkiableTerrain_ac	3	0.909091
yearsOpen	1	0.303030
total_chairs	0	0.000000

fast eight has the most missing values, at just over 50%. also missing quite a few of our desired target quantity, the ticket price, which is missing 15-16% of values. AdultWeekday is missing in a few more records than AdultWeekend.

Categorical features

- Name , Region, state: These three columns had no missing values
- Name isn't unique, but combination of (Name+Region)/(Name+state) Is.
- Region is NOT always the same as state

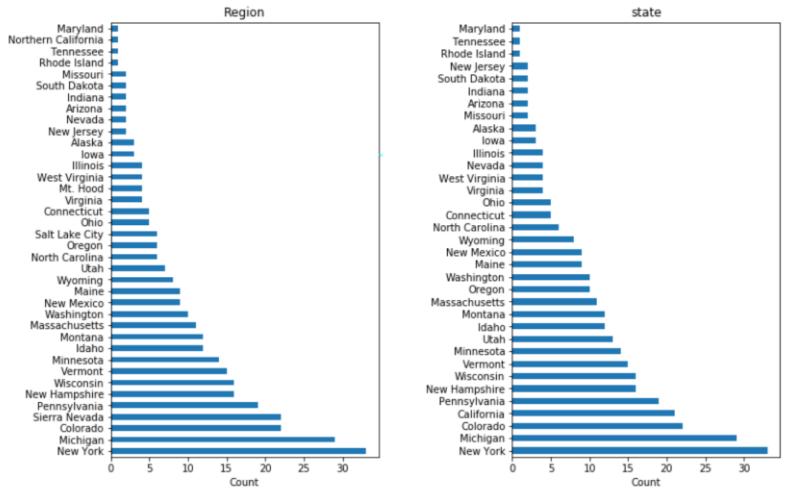
There are two Crystal Mountain resorts, but they are clearly two different resorts in two different states. This is a powerful signal that we have unique records on each row.

Relationship between Region and state

A casual inspection by eye reveals some non-state names such as Sierra Nevada, Salt Lake City, and Northern California. Tabulate the differences between Region and state.

state	Region	
California	Sierra Nevada	20
Northern California	1	
Nevada	Sierra Nevada	2
Oregon	Mt. Hood	4
Utah	Salt Lake City	6

The vast majority of the differences are in California, with most Regions being called Sierra Nevada and just one referred to as Northern California.



New York accounting for the majority of resorts. Our target resort is in Montana, which comes in at 13th place.

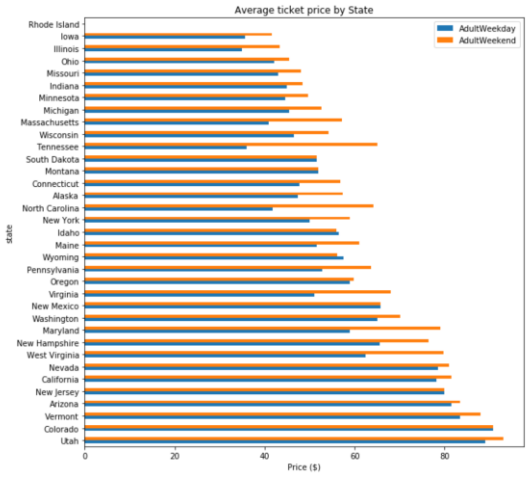
Should I create Montana-specific Model or All- state model?

(?) Does New York command a premium because of its proximity to population? Even if a resort's State were a useful predictor of ticket price, our main interest lies in Montana. Would we want a model that is skewed for accuracy by New York? Should we just filter for Montana and create a Montana-specific model? This would slash the available data volume (to 12 resorts' records).

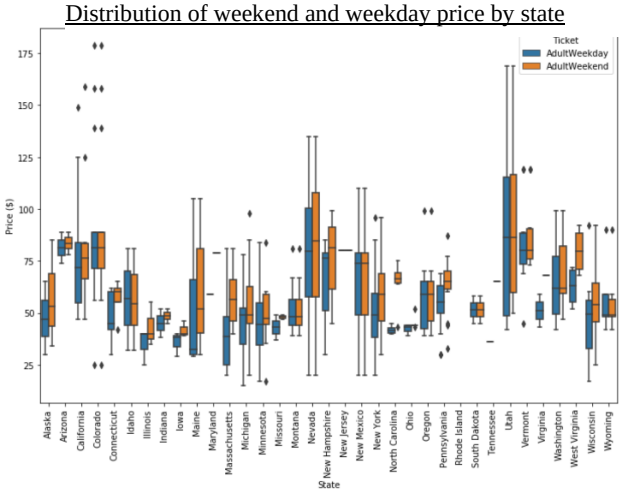
Problem task includes the contextual insight that the data are for resorts all belonging to the same market share. This suggests one might expect prices to be similar amongst them. I will look into this, with a boxplot grouped by State is an ideal way to quickly compare prices.

Another side note worth bringing up here is that, in reality, the best approach here definitely would include consulting with the client or other domain expert. They might know of good reasons for treating states equivalently or differently.

Distribution of average ticket price by state



Distribution of weekend and weekday price by state



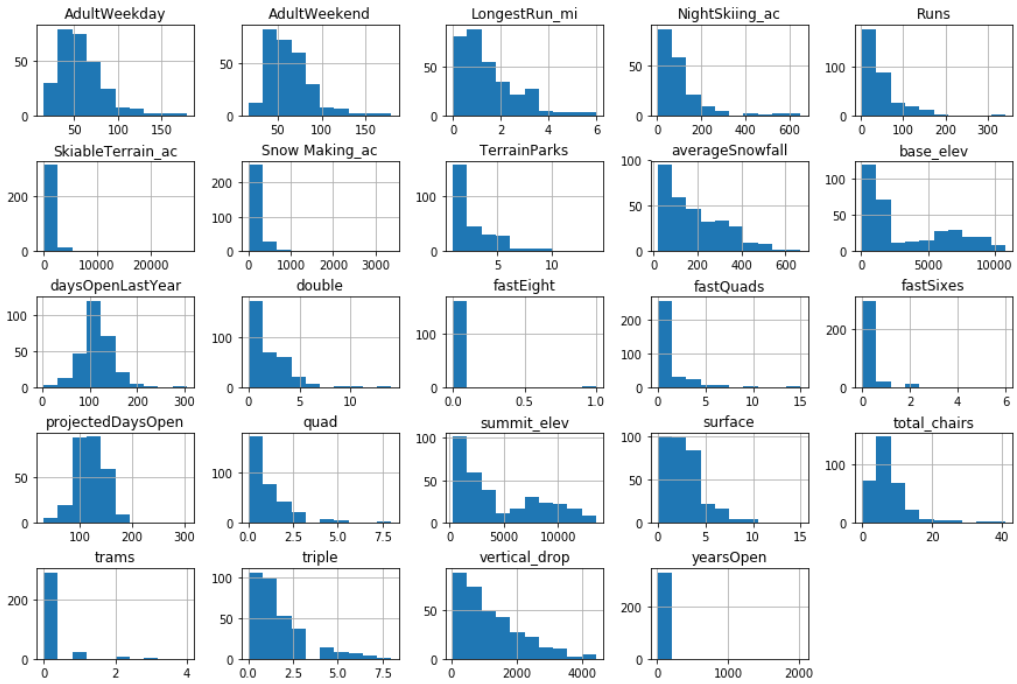
Aside from some relatively expensive ticket prices in California, Colorado, and Utah, most prices appear to lie in a broad band from around 25 to over 100 dollars. Some States show more variability than others. Montana and South Dakota, for example, both show fairly small variability as well as matching weekend and weekday ticket prices. Nevada and Utah, on the other hand, show the most range in prices. Some States, notably North Carolina and Virginia, have weekend prices far higher than weekday prices. Although I could be inspired from this exploration to consider a few potential groupings of resorts, those with low spread, those with lower averages, and those that charge a premium for weekend tickets. However, as mentioned before, I was told that i should take all resorts to be part of the same market share, therefore I'll decide to retain all State information

Numeric features

	count	mean	std	min	25%	50%	75%	max
summit_elev	330.0	4591.818182	3735.535934	315.0	1403.75	3127.5	7806.00	13487.0
vertical_drop	330.0	1215.427273	947.864557	60.0	461.25	964.5	1800.00	4425.0
base_elev	330.0	3374.000000	3117.121621	70.0	869.00	1561.5	6325.25	10800.0
trams	330.0	0.172727	0.559946	0.0	0.00	0.0	0.00	4.0
fastEight	164.0	0.006098	0.078087	0.0	0.00	0.0	0.00	1.0
fastSixes	330.0	0.184848	0.651685	0.0	0.00	0.0	0.00	6.0
fastQuads	330.0	1.018182	2.198294	0.0	0.00	0.0	1.00	15.0
quad	330.0	0.933333	1.312245	0.0	0.00	0.0	1.00	8.0
triple	330.0	1.500000	1.619130	0.0	0.00	1.0	2.00	8.0
double	330.0	1.833333	1.815028	0.0	1.00	1.0	3.00	14.0
surface	330.0	2.621212	2.059636	0.0	1.00	2.0	3.00	15.0
total_chairs	330.0	8.266667	5.798683	0.0	5.00	7.0	10.00	41.0
Runs	326.0	48.214724	46.364077	3.0	19.00	33.0	60.00	341.0
TerrainParks	279.0	2.820789	2.008113	1.0	1.00	2.0	4.00	14.0
LongestRun_mi	325.0	1.433231	1.156171	0.0	0.50	1.0	2.00	6.0
SkiableTerrain_ac	327.0	739.801223	1816.167441	8.0	85.00	200.0	690.00	26819.0
Snow Making_ac	284.0	174.873239	261.336125	2.0	50.00	100.0	200.50	3379.0
daysOpenLastYear	279.0	115.103943	35.063251	3.0	97.00	114.0	135.00	305.0
yearsOpen	329.0	63.656535	109.429928	6.0	50.00	58.0	69.00	2019.0
averageSnowfall	316.0	185.316456	136.356842	18.0	69.00	150.0	300.00	669.0
AdultWeekday	276.0	57.916957	26.140126	15.0	40.00	50.0	71.00	179.0
AdultWeekend	279.0	64.166810	24.554584	17.0	47.00	60.0	77.50	179.0
projectedDaysOpen	283.0	120.053004	31.045963	30.0	100.00	120.0	139.50	305.0
NightSkiing_ac	187.0	100.395722	105.169620	2.0	40.00	72.0	114.00	650.0

Just over 82% of resorts have no missing ticket price, 3% are missing one value, and 14% are missing both. I will drop the records for which th is no price information, but only after recieving a fuller pictire for rest of the features.

Distribution of numeric features

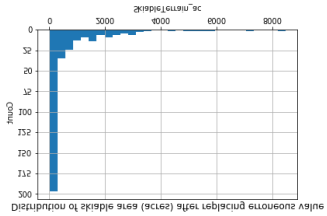


Some of the following features are possibly a cause for concern:
SkiableTerrain_ac - because values are clustered down the low end,
Snow Making_ac - for the same reason,
fastEight - because all but one value is 0 so it has a minimal variance, and half the values are missing,
fastSixes - raises an amber 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, it has a maximum of 2019, which strongly suggests someone recorded calendar year rather than the number of years.

Dealing with suspicious records:

Silverton Mountain Resort has 26,819 skiable terrain in ac:

According to this useful information on their site (https://silvertonmountain.com/mountain/), I found that the real value is 1,819 ac.



Although it is still a long-tailed distribution, most extreme value now is slightly above 8000, which make this above distribution more plausible.

Heavenly Mountain Resort has 3,379 Snow making area:

In their website (https://www.skiheavenly.com/the-mountain/about-the-mountain/mountain-info.aspx), We see that we have values for skiable terrain that agree also it is mentioned that **snowmaking covers 60% of the trails**

After calculating the area covered by snowmaking, we see it is less than in our data. 2,880 ac (=4,800*0.6) as apose to 3,379 ac. This is less than the value of 3379 in our data. We can adjust it, however, there is no ticket pricing information at all for this resort. Any further effort spent about the values for this resort will be wasted. I'll simply be dropping the entire row.

Number of missing values by column

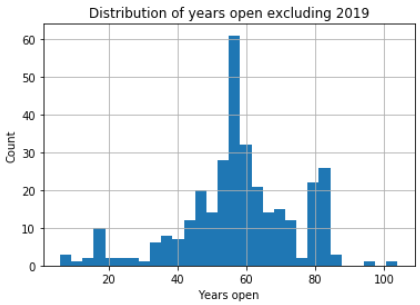
fast Eight column

```
0.0    163
1.0      1
```

Half the values are missing and all but one zeros. There is essentially no information in this column. so it will be dropped entirety.

years open column

One seems to have been open for 104 years and another for 2019 years. This is most likely wrong.



The above distribution of years seems entirely plausible, including the 104 year value. We can certainly state that no resort will have been open for 2019 years It likely means the resort opened in 2019 or it could also mean the resort is due to open in 2019 (as we don't know when these data were gathered).

Summary statistics for yearsOpen excluding ‘2019’:

```
count    328.000000
mean      57.695122
std       16.841182
min        6.000000
25%       50.000000
50%       58.000000
75%       68.250000
max      104.000000
```

From the above summary statistics, the smallest number of years open otherwise is 6. We can't be sure whether this resort in question has been open zero years or two years and even whether the numbers are projections or actual. In any case, we won't be adding a new youngest resort so it best to drop this row.

fastSixes and Trams
The other features had mild concern over, so I will leave the data as it is.

Derive state-wide summary statistics

By this point one row was removed for a resort that may not have opened yet (or in its firsts seasons). Basic business knowledge tells us that state-wide supply and demand of certain skiing resources may well factor into pricing strategies. Does our resort dominate some area?

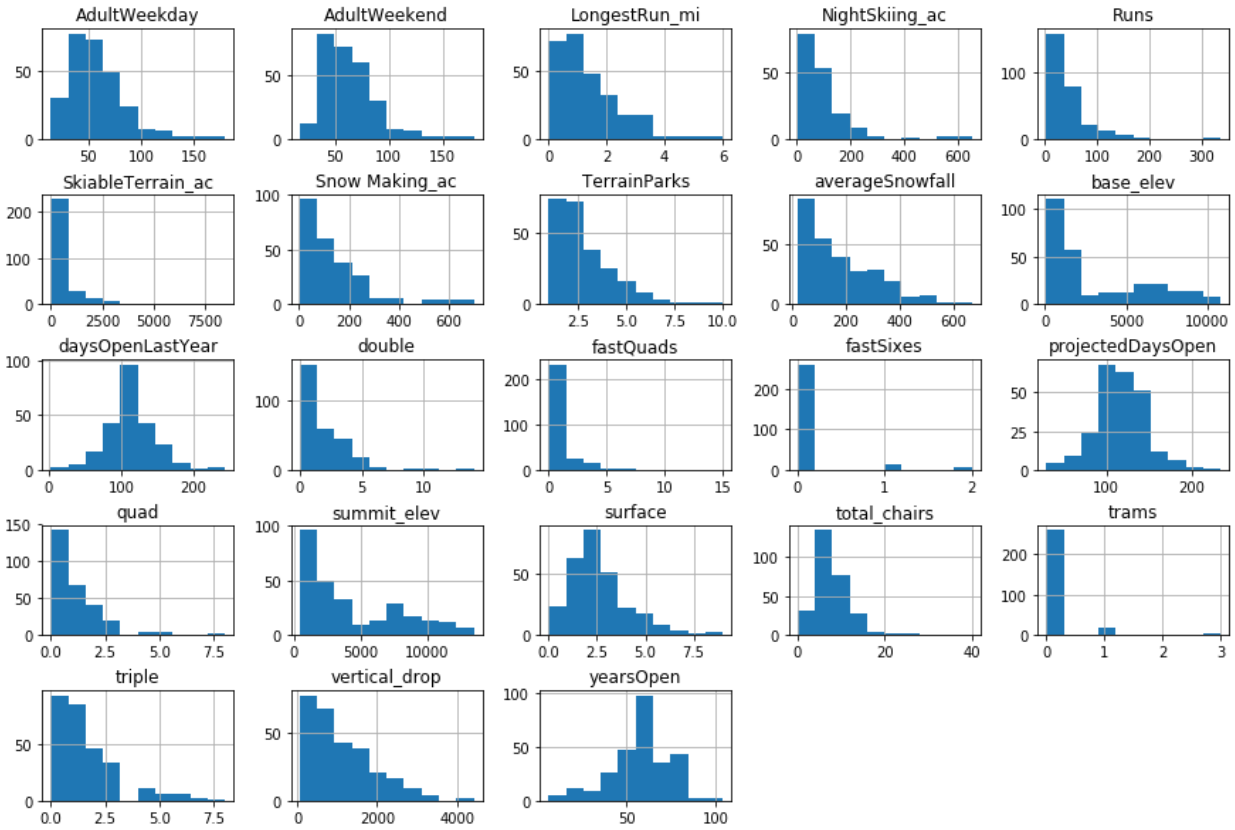
We should use all feature records of the data that can capture the state-wide market statistics, **before** dropping any more rows.
It makes sense to sum these features to to captures the state-wide market size: the total number of terrain parks, the total skiable area, the total number of days open, and the total area available for night skiing.
Note: We might consider the total number of ski runs, but perhapse skiable area is more informative than just the number of runs.

Adding to the dataFrame the following columns (by state-wise aggregation:sum) :

- resorts_per_state
- state_total_skiable_area_ac
- state_total_days_open
- state_total_terrain_parks
- state_total_night_skiing_ac

Dropping all rows with missing pricing information

There are two columns that refer to price: 'AdultWeekend' and 'AdultWeekday'. About 14% of the rows have no price data. As the price is our target, these rows are of no use and will be dropped. **47 records were dropped.**

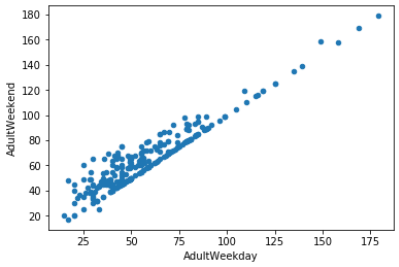


After dropping a total of 49 records, distributions are much better. There are some skewed distributions, so it's worth to keep an eye on `fastQuads`, `fastSixes`, and `trams`. These lack much variance away from 0 and may have a small number of relatively extreme values.

State-Population and State-Area data

Population and area data for the US states was obtained from wikipedia (https://simple.wikipedia.org/wiki/List_of_U.S._states). with this data we can proceed with an analysis that includes state sizes and populations for the 'first cut' model.
later I will join this with the ski resort data to explore the data, including the relationships between the states.

Target feature – Weekend Vs. Weekday



	AdultWeekend	AdultWeekday
141	42.0	42.0
142	63.0	63.0
143	49.0	49.0
144	48.0	48.0
145	46.0	46.0
146	39.0	39.0
147	50.0	50.0
148	67.0	67.0
149	47.0	47.0
150	39.0	39.0
151	81.0	81.0

←prices for Montana resorts only

There is a clear line where weekend and weekday prices are equal. Weekend prices being higher than weekday prices seem restricted to sub $100 resorts. In Montana weekday and Weekens ticket prices are equal, therfore there isn't a reason to prefer weekend or weekday prices, so it's better to choose the one which is missing the least.
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.

GUIDED CAPSTONE PROJECT REPORT – Libi Voshin

SUMMARY OF DATA WRANGLING PHASE

- We started with 330 resort records, and 27 columns, with "big Mountain Resort" among this data
- During the data cleaning process, some feature columns as well as some record lines were removed. these include:
  1. fastEight* column- half the values are missing, and all but one resort having zero 'fastEight' seat. There is essentially no information in this column
  2. Weekday* column- Weekend prices have the least missing values of the two (4 vs 7 missing) therefore we chose to drop the weenday prices column.
  3. data record of "Pine Knob Ski Resort" in Michigan, was removed due to it's unclear 'Years Open' data(=2019). No resort will have been open for 2019 years, and beacause we don't know when this data was gathered, we can say whether it has been open zero or two years. Anyway the smallest number of years open otherwise is 6, so we decided not to consider resort that may not have been opened yet, or perhaps in its first season.
  4. After deriving all state-wide statistics, we dropped all rows with missing price information, that were about 14.33% of the rows. (48 rows, including previosly removed record of 'Heavenly Mountain Resort' that was removed during 'snow making_ac' outlier value inspection.)

- Other issue found, Skiable erea in ac caused a concern because values were clustered down the low end. after further exploring, Silverton Mountain Resort had highly suspicious 26,819 skiable terrain in ac, the value I've looked up in other data scorce was 1819 ac. this led to the data correction step: the suspect value was replaced with the new one I've obtained.

- By knowing that state-wide supply and demand of certain skiing resources may well factor into pricing strategies, We derived state-wide summary statistics to try and answer whether our resort dominates certain domain?

  - State-wide derived data includes:
    1. resorts_per_state
    2. state_total_skiable_area_ac
    3. state_total_days_open
    4. state_total_terrain_parks
    5. state_total_night_skiing_ac

- At this stage of the project, we left with 227 resort recods to work on. And the data science problem I subsequently identified is to predict the adult weekend ticket price for ski resorts.

Take aways during – Exploratory data analysis

Top States By Order Of Each Of The Summary Statistics

Total state erea – Montana comes in at third largest
Alaska 665384
California 163695
Montana 147040
New Mexico 121590
Arizona 113990

Total state population – Montana cones 26th , not densely populated at all
California 39512223
New York 19453561
Pennsylvania 12801989
Illinois 12671821
Ohio 11689100

Resorts per state – Montana 11th , with 12 resorts
New York 33
Michigan 28
Colorado 22
California 21
Pennsylvania 19

Total skiable erea – Montana 4th
Colorado 43682.0

Utah 30508.0
California 25948.0
Montana 21410.0
Idaho 16396.0

- New York state may have the most resorts, but they don't account for the most skiing area.
- Montana makes it into the top five.
- New York has more, smaller resorts, whereas Montana has fewer, larger resorts.
- Colorado seems to have a name for skiing; it's in the top five for resorts and in top place for total skiable area.

Total night skiing erea ac – Montana 8th
New York 2836.0
Washington 1997.0
Michigan 1946.0
Pennsylvania 1528.0
Oregon 1127.0

New York dominates the area of skiing available at night.
Looking at the top five in general, they are all the more northerly states. Is night skiing in and of itself an appeal to customers, or is a consequence of simply trying to extend the skiing day where days are shorter? Is New York's domination here because it's trying to maximize its appeal to visitors who'd travel a shorter distance for a shorter visit? You'll find the data generates more (good) questions rather than answering them. This is a positive sign. We might ask our executive sponsor or data provider for some additional data about typical length of stays at these resorts, although we might end up with data that is very granular and most likely proprietary to each resort. A useful level of granularity might be "number of day tickets" and "number of weekly passes" sold.

Total days open – Montana 15th (open 951 days)
Colorado 3258.0
California 2738.0
Michigan 2389.0
New York 2384.0
New Hampshire 1847.0

The total days open seem to bear some resemblance to the number of resorts. This is plausible. The season will only be so long, and so the more resorts open through the skiing season, the more total days open we'll see. New Hampshire makes a good effort at making it into the top five, for a small state that didn't make it into the top five of resorts per state. Does its location mean resorts there have a longer season and so stay open longer, despite there being fewer of them?

Resort density

There are big states which are not necessarily the most populous. There are states that host many resorts, but other states host a larger total skiing area. The states with the most total days skiing per season are not necessarily those with the most resorts. And New York State boasts an especially large night skiing area. New York had the most resorts but wasn't in the top five largest states, so the reason for it having the most resorts can't be simply having lots of space for them. New York has the second largest population behind California. Perhaps many resorts have sprung up in New York because of the population size? Does this mean there is high competition between resorts in New York State, fighting for customers and thus keeping prices down? We're not concerned with the absolute size or population of a state, but we could be interested in the ratio of resorts serving a given population or a given area.

Adding to the dataFrame the following columns (by state-wise aggregation:sum) :

- resorts_per_100kcapita
- resorts_per_100ksq_mile
- state_population

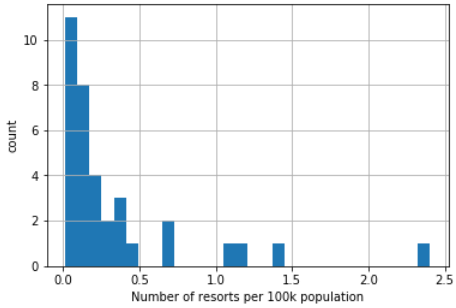
Note: 100,000 scalings is simply based on eyeballing the magnitudes of the data

With the removal of the two columns that only spoke to state-specific data, now the Dataframe speaks to the skiing competitive landscape of each state. It has the number of resorts per state, total skiable area, and days of skiing. We've translated the plain state data into useful data that gives an idea of the density of resorts relative to the state population and size

resort density (resorts_per_100kcapita) – Montana 4th

Vermont 2.403889
Wyoming 1.382268
New Hampshire 1.176721
Montana 1.122778
Idaho 0.671492

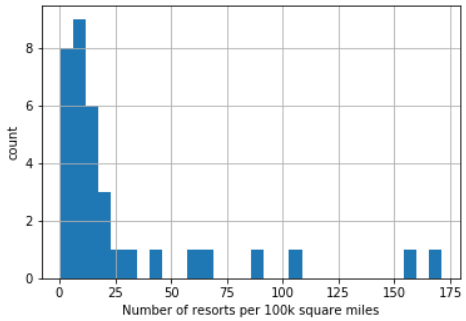
Resorts per 100K capita distribution



resort density (resorts_per_100ksq_mile) – Montana 24th

New Hampshire 171.141299
Vermont 155.990017
Massachusetts 104.225886
Connecticut 90.203861
Rhode Island 64.724919

Resorts per 100K square-miles distribution



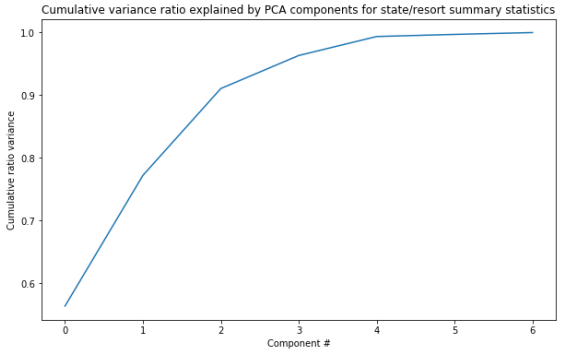
Vermont seems particularly high in terms of resorts per capita, and both New Hampshire and Vermont top the chart for resorts per area. New York doesn't appear in either.

Visualizing High Dimensional Data – with PCA

Some states are higher in some but not in others. Some features also are more correlated with one another than others. One way to disentangle this interconnected web of relationships is via PCA.
This technique will find linear combinations of the original features that are uncorrelated with one another and order them by the amount of variance they explain.
Then, I could visualize the data in a **lower dimension** structure and know how much variance the representation explains.

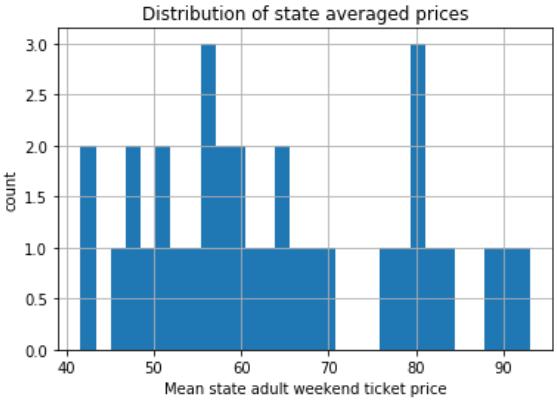
The basic steps in this process are:
1. scaling the data (important because our features are heterogeneous)
2. fitting the PCA transformation (learn the transformation from the data)

3. applying 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



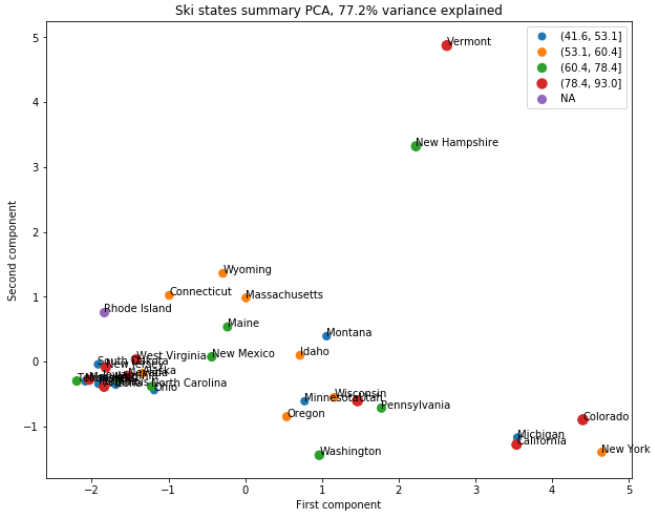
The first two components seem to account for over 75% of the variance, and the first four for over 95%.

Adding average ticket price by state



To better distinct ticket price in the following plotting, of first and second principal components of each state, I added another column where these prices will be separeted into 4 quartiles:

1st quartile: Prices between $41.6 - $53.1 [41.6 , 53.1]
2nd quartile: Prices between $53.1 - $60.4 [53.1 , 60.4]
3rd quartile: Prices between $60.4 - $78.4 [60.4 , 78.4]
4th quartile: Prices between $78.4 - $93.0 [78.4 , 93.0]



In this representation of the ski summaries for each state, which accounts for some 77% of the variance: There isn't an obvious pattern.
The red points, representing the upper quartile of price, can be seen to the left, the right, and up top. There's also a spread of the other quartiles as well.
In the first two components, there is a spread of states across the first component.
It looks like Vermont and New Hampshire might be off on their own a little in the second dimension, although they're no more extreme than New York and Colorado are in the first dimension. The `components` attribute of the fitted PCA object tell us how important (and in what direction) each feature contributes to each score.

`resorts_per_100kcapita` and `resorts_per_100ksq_mile` might count for quite a lot here. and yes, both states have particularly large values of `resorts_per_100ksq_mile` in absolute terms, and these put them more than 3 standard deviations from the mean. Vermont also has a notably large value for `resorts_per_100kcapita`. New York, then, does not seem to be a stand-out for density of ski resorts either in terms of state size or population count.

Summary

This process offers some justification for treating all states equally, and work towards building the pricing model that considers all states together, without treating any one particularly specially.
We havn't seen any clear grouping yet but captured potentially relevant state data in features most likely to be relevant to our business use case.

Putting each resort within the context of its state

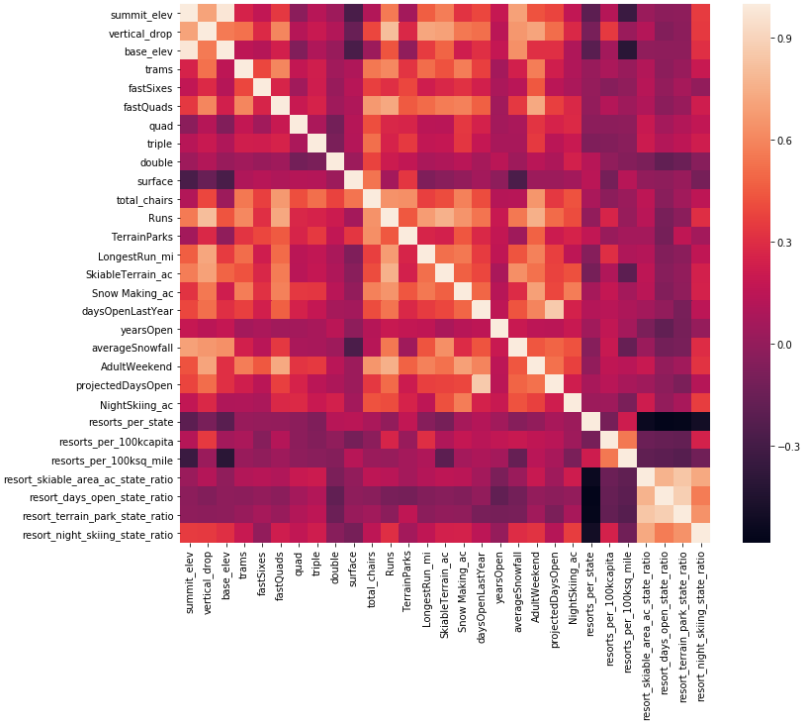
Having merged our state summary features into the ski resort data, we added following "state resort competition" features:

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

Feature correlation heatmap

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



Heatmap main take aways:

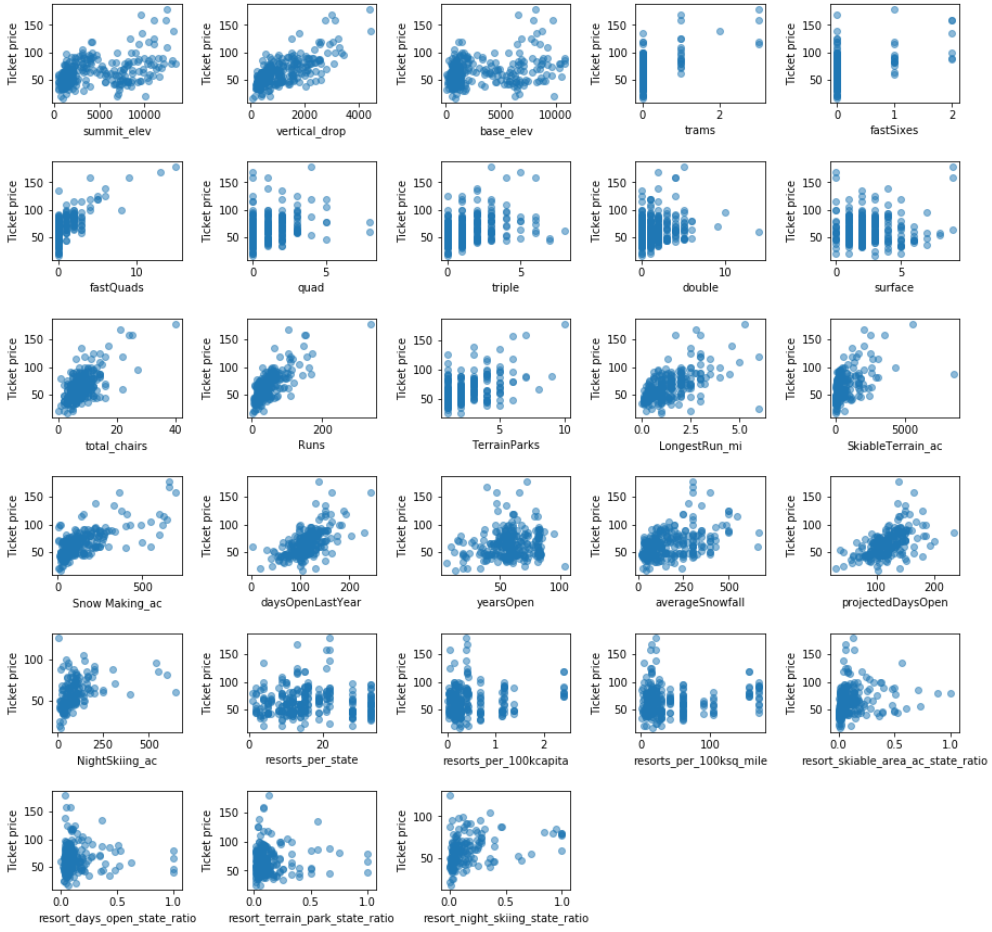
1. summit and base elevation are quite highly correlated. This isn't a surprise.
2. We also introduced a lot of multicollinearity with our new ratio features; they are negatively correlated with the number of resorts in each state. This makes sense, If we increase the number of resorts in a state, the share of all the other state features will drop for each.
3. There is some positive correlation between the ratio of night skiing area with the number of resorts per capita. In other words, it seems that when resorts are more densely located with population, more night skiing is provided.

Turning the attention to the target feature, `AdultWeekend` ticket price, We see quite a few reasonable correlations:

1. `**fastQuads**` stands out, along with `**Runs**` and `**Snow Making_ac**` (visitors would seem to value more guaranteed snow, which would cost in terms of snow making equipment, which would drive prices and costs up).
2. Of the new features, `r**esort_night_skiing_state_ratio**` seems the most correlated with ticket price. If this is true, then perhaps seizing a greater share of night skiing capacity is positive for the price a resort can charge.
3. `**Runs**`, `**total_chairs**` are also quite well correlated with ticket price. The more runs resort have, the more chairs it need to ferry people to them. Interestingly, these count more than the total skiable terrain area.
   * perhapse the total skiable terrain area is not as useful as the area with snow making. People seem to put more value in guaranteed snow cover rather than more variable terrain area.
4. The `**vertical drop**` seems to be a selling point that raises ticket prices as well.

Scatterplots of numeric features against ticket price

It's helpful to see correlations together in a heatmap to identifying patterns but correlation can mask relationships between two variables. next I will create a series of scatterplots to really dive into how ticket price varies with other numeric features.



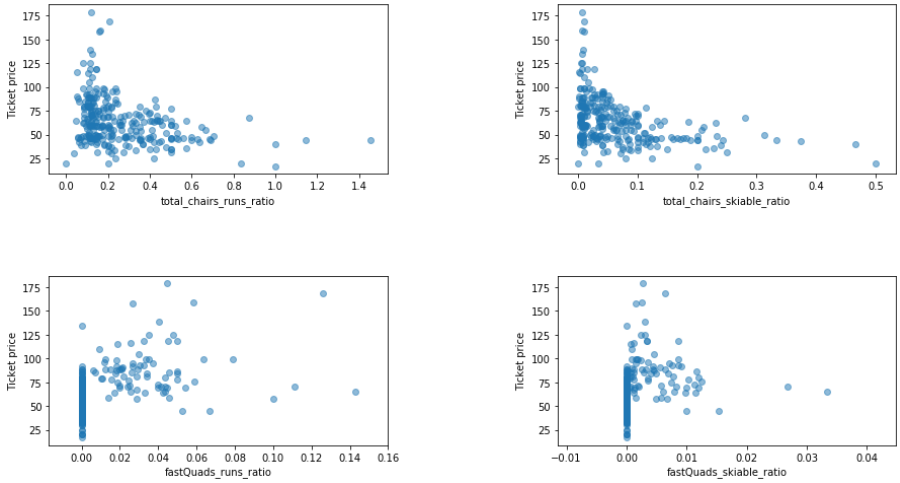
In the scatterplots some of the high correlations were clearly picking up on. There's a strong positive correlation with `vertical_drop`. `fastQuads` seems very useful. `Runs` and `total_chairs` appear quite similar and also useful. `resorts_per_100kcapita` shows something interesting, When the value 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.

Exploring How easily a resort can transport visitors around the resort

Further features that may be useful, is **how easily a resort can transport people around**. We have the numbers of various chairs, and the number of runs, so we can have the ratio of chairs to runs. It seems logical that this ratio would inform us how easily and quickly, people could get to their next ski slope.

Deriving the following feature columes:

- 'total_chairs_runs_ratio'
- 'total_chairs_skiable_ratio',
- 'fastQuads_runs_ratio'
- 'fastQuads_skiable_ratio



Take aways:
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 a resort doesn't have so many chairs, it can charge more for its tickets, although with fewer chairs resort beeing able to serve fewer visitors, Its price per visitor is high but the number of visitors may be low. Useful data that's missing is the number of visitors in each resort per year.
It also appears that having no fast quads may limit the ticket price, but if a resort covers a wide area then getting a small number of fast quads may be beneficial to ticket price.

SUMMARY OF EDA PHASE

By knowing that state-wide supply and demand of certain skiing facilities may factor into pricing strategies, We derived state-wide summary statistics to try and answer whether our resort dominates certain domain?

State-wide derived *numeric features* included:

1. resorts_per_state
2. state_total_skiable_area_ac
3. state_total_days_open
4. state_total_terrain_parks
5. state_total_night_skiing_ac

Additionally, obtained from wikipedia source :

6. state_population - > resorts per 100K capita
7. state_area (sq miles) -> resorts per 100L sq miles

state-wise we haven't seen any clear grouping or pattern suggested of a relationship between state and ticket price.

After diving into resort-level, our data *numeric features* included:

1. resorts_per_state
2. resort_skiable_area_ac_state_ratio
3. resort_days_open_state_ratio
4. resort_terrain_park_state_ratio
5. resort_night_skiing_state_ratio
6. resorts per 100K capita
7. resorts per 100L sq miles

Another feature that may be useful is how easily a resort can transport people around. from numbers of chairs and runs, we formed the following numeric features:

8. total_chairs_runs_ratio
9. total_chairs_skiable_ratio
10. fastQuads_runs_ratio
11. fastQuads_skiable_ratio

We saw a strong positive correlation between: fastQuads, Runs,Snow Making_ac, total_chairs, vertical drop and ticket price.
But correlation relationships between the variables themselves could exist as well and we should remain wary of when performing feature selection for modeling.
For our model we want to be looking at all the given and derived numeric features available.

Take aways during – Data preprocessing and training

In this phase I'll start to build machine learning models.
The first model is a baseline performance comparitor for any subsequent model, we will use the mean value as a predictor, We then build up the process of efficiently and robustly creating and assessing models against it.

Train/Test split 70/30

In machine learning, by partitioning the data into training and testing splits, without letting a model (or missing-value imputation) learn anything about the test split, we get somewhat independent assessment of how the model might perform in the future.

We split the data to: **70% train (193 records) / 30% test (83 records).**

To avoid a frequently use of the TEST split to assess model performance (and then compare multiple models to pick the best), meaning OVERFITTING the model to one specific data set (the TEST split). We will use the **cross-validation** while training models. So that the TEST split will be useful as a final check on expected future performance.

First we'll explore
* TRAIN split: {trained_Y, predicted_trained_Y}
* TEST split: {test_Y, predicted_test_Y}

Then, asses performance with built-in Sklearn Metrics:
1. r2_score(train_y, predicted_y)
2. mean_absolute_error(y_train, predicted_y)
3. mean_squared_error(y_train, predicted_y)

Initial Method details:

1. Imputing missing feature (predictor) values

    1.1 impute with median

from our data exploration that many distributions were skewed. Our first thought might be to impute missing values using the median, These are the values we'll use to fill in any missing value

1.1.1 claculate median from Train set to all features/columns
X_defaults_median = X_train.median()

1.1.2 Apply the imputation to both train and test splits

X_tr = X_train.fillna(X_defaults_median)

X_te = X_test.fillna(X_defaults_median)

2. Scale data - The StandardScaler scales each feature to zero mean and unit variance.
3. Train the model on the train split

For example **Simple Linear regression model**:
lm = LinearRegression().fit(X_tr_scaled, y_train)

4. Make predictions using the model on both train and test splits

y_tr_pred = lm.predict(X_tr_scaled)
y_te_pred = lm.predict(X_te_scaled)

5. Assess model performance (on Train & Test)

with R^2 , MAE, MSE

1. Replace/impute missing values
2. scale the data (to zero mean and unit variance)
3. train a model
4. calculate model performance

All these steps were trained on the train split and then applied to the test split for assessment.

These are common steps and sklearn provides easy way to use them, with Pipelines.

6. **Pipelines**

We can wrap the entire process of imputing and feature scaling and regression in a single object we can train with .fit() and predict with .predict(). And that's basically a pipeline: a model on steroids.

6.1. **Define the Pipeline**

```
pipe = make_pipeline(
   SimpleImputer(strategy='median'),
   StandardScaler(),
   LinearRegression()
)
```

type(pipe) - sklearn.pipeline.Pipeline

6.2. **Fit the pipeline**

```
pipe.fit(X_train, y_train)
```

Here, a single call to the pipeline's fit() method combines the steps of learning the imputation (determining what values to use to fill the missing ones), the scaling (determining the mean to subtract and the variance to divide by), and then training the model. It does this all in the one call with the training data as arguments

6.3. **Make Predictiona on the train and test sets**

```
y_tr_pred = pipe.predict(X_train)
y_te_pred = pipe.predict(X_test)
```

6.4. **Assess performance**

7. Refining The Linear Model (Optional)

When suspected that the model is overfitting. (no surprise given the number of features we blindly used). It's likely a judicious subset of features would generalize better.

sklearn has a number of feature selection functions available

7.1. SelectKBest, selects the k best features here.
f_regression is just the score function you're using because you're performing regression. It's important to choose an appropriate one for our machine learning task.

7.1.1 Redefine our pipeline to include this feature selection step. Simply by adding:
SelectKBest(f_regression)
Inside the: make_pipeline()

7.1.2 Continue usual steps: (fit, assess performance

We 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.
But it will lead to a model that works well on the quirks of our test set *but fails to generalize to new data*.

The way around this is *cross-validation.*

8. Assesing performance using **Cross-Validation**

8.1. We partition the training set into k folds, (cv=folds number) train 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 we 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.

```
cv_results = cross_validate(desired_pipe, X_train, y_train, cv=5)
cv_scores = cv_results['test_score']
np.mean(cv_scores), np.std(cv_scores)
```

8.2. Model performance in inherently open to variability. You'll get different results depending on the quirks of which points are in which fold. An advantage of this is that we can also obtain an estimate of the variability, or uncertainty, in our performance estimate.

```
np.round((np.mean(cv_scores) - 2 * np.std(cv_scores), np.mean(cv_scores) + 2 *
np.std(cv_scores)), 2)
```

Pulling the above together, we have:

- a pipeline that
  - imputes missing values
  - scales the data
  - selects the k best features
  - trains a linear regression model
- a technique (cross-validation) for estimating model performance

9. Hyperparameter search using **GridSearchCV**

{ Model parameters are estimated from data automatically

Model hyperparameters are set manually and are used in processes to help estimate model parameters. Model hyperparameters are often referred to as parameters because they are the parts of the machine learning that must be set manually and tuned. }

make_pipeline automatically names each step as the lowercase name of the step and the parameters of the step are then accessed by appending a double underscore followed by the parameter name. We know the name of the step will be 'selectkbest' and we know the parameter is 'k'.

We can also list the names of all the parameters in a pipeline like this:

```
pipe.get_params().keys()
```

for example:
'selectkbest__k' will have an array/list of all range parameters from 1 to columns(/features) number

Now we use cross-validation for multiple values of k and use cross-validation to pick the value of k that gives the best performance.

GridSearchCV iterate over each possible value, to track the best value of k.

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, we 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.

9.1 Perform GridSearchCV

```
lr_grid_cv = GridSearchCV(desired_pipe, param_grid=grid_params, cv=5, n_jobs=-1)
```

9.2 Fit model

9.3 Get scores of: 'mean_test_score' & 'std_test_score'

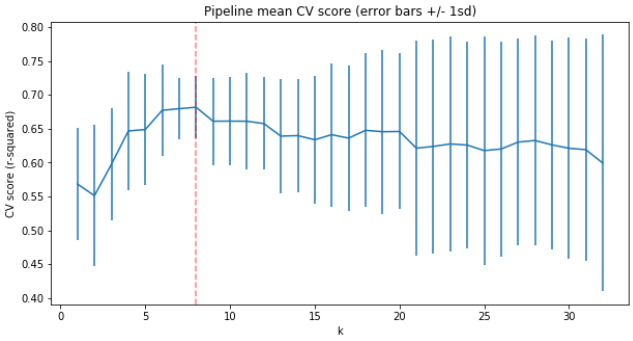
```
score_mean = lr_grid_cv.cv_results_['mean_test_score']

score_std = lr_grid_cv.cv_results_['std_test_score']
```

9.4 Get best K – by getting `best_params_` attribute of `lr_grid_cv`

```
lr_grid_cv.best_params_
```

9.5 Plotting an errorbar of the Pipeline mean CV score (+/- 1 std):



The above suggests a good value for k is 8. 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 we increasingly overfit, we should expect greater swings in performance as different points move in and out of the train/test folds.

9.6 Get the list of K features that were most useful.
access the named step for the linear regression model and, from that, grab the model coefficients via its coef_ attribute:

```
coefs = lr_grid_cv.best_estimator_.named_steps.linearregression.coef_
features = X_train.columns[selected]
```

```
pd.Series(coefs, index=features).sort_values(ascending=False)
```

"Best" K features are:

```
vertical_drop        10.767857
Snow Making_ac        6.290074
total_chairs          5.794156
fastQuads             5.745626
Runs                  5.370555
LongestRun_mi         0.181814
trams                -4.142024
SkiableTerrain_ac    -5.249780
```

Main take aways:

These results suggest that vertical drop is our biggest positive feature. Also, the area covered by snow making equipment is a strong positive as well. Those are consistent with what we saw during the EDA work.
The skiable terrain area is negatively associated with ticket price. People will pay less for larger resorts, There could be all manner of reasons for this. It could be an effect whereby larger resorts can host more visitors at any one time and so can charge less per ticket. As has been mentioned previously, the data are missing information about visitor numbers.

Random Forest Model

there's no need to check performance on the test split. Instead, cross_validate will perform the fitting on both train and test splits, as part of the process. This uses the default settings for the random forest

Methods:

apply(X)	Apply trees in the forest to X, return leaf indices.
decision_path(X)	Return the decision path in the forest.
fit(X, y[, sample_weight])	Build a forest of trees from the training set (X, y).
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict regression target for X.
score(X, y[, sample_weight])	Return the coefficient of determination R2 of the prediction.
set_params(**params)	Set the parameters of this estimator.

14.1 Define the pipeline

```
RF_pipe = make_pipeline(
    SimpleImputer(strategy='median'),
    StandardScaler(),
    RandomForestRegressor(random_state=47)
)
```

14.2 Fit pipline

14.3 Assess performance using Cross-Validation

14.4 Hyperparameter search using GridSearchCV

Random forest has several hyperparameters that can be explored. However in this phase I had limited to following values:
- The number of trees.
- With and without feature scaling
- Trying both the mean and median as strategies for imputing missing values.

Encouragingly, the dominant top four features are in common with the linear model:
* fastQuads
* Runs
* Snow Making_ac
* vertical_drop

Final model selection

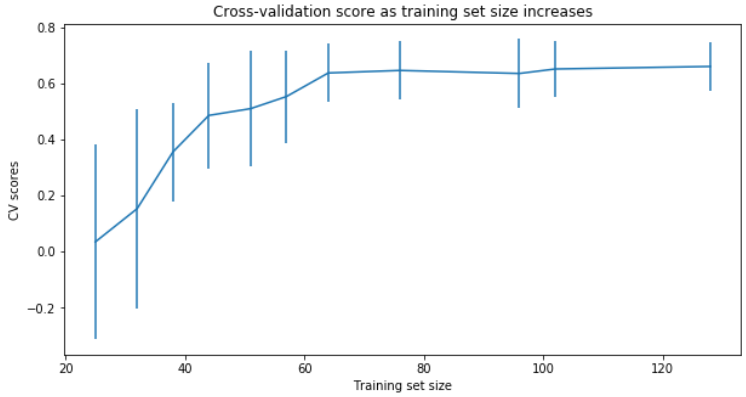
We built a best linear model and a best random forest model. we need to finally choose between them.
To compare performance, we calculated the mean absolute error using cross-validation:

Performance	(mean , std) of all 5 MAE scores, on Train st	MAE on TEST split
Linear Regression	(10.499, 1.622)	11.793
Random Forest	(9.659, 1.350)	9.4955

Conclusion:
The random forest model has a lower cross-validation mean absolute error by almost $1. It also exhibits less variability. Verifying performance on the test set produces performance consistent with the cross-validation results.

Data quantity assessment

To advise the business whether it needs to undertake further data collection, we'll assess this by seeing how performance varies with differing data set sizes. (using the `learning_curve` function).



This shows that there seems to be plenty of data. There's an initial rapid improvement in model scores, but it's essentially levelled off by around a sample size of 60.

SUMMARY OF PREPROCESSING AND TRINING PHASE

We started with a baseline idea of performance by simply taking the average price of the train set, \\$63.811 asessing performance of this prediction on the test set. -0.312% is the amount of variance explained (R^2) with the mean as a predictor.
Mean Absolute Error, that summarise the difference between predicted and actual values was 17.9235 on train set and 19.136 on the Test set.

• We built a best linear model and a best random forest model
1. Linear model - Imputing missing values with Median, scaling the data to zero mean and one std, and selecting 8 best following features:

vertical_drop, Snow Making_ac, total_chairs, fastQuads, Runs, LongestRun_mi, trams, skiableTerrain_ac

MAE performance from Cross-Validation on the Train set was 10.499 with std of 1.622, Verifying performance on the test set produces performance of 11.792, consistent with the cross-validation results.

2. Random Forest model - also imputing missing values with Median, without data scaling.

MAE performance from Cross Validation was slightly better 9.659 with lower std (1.349) on train set and 9.495 MAE on the Test set.

Conclusions:
The random forest model has a lower cross-validation mean absolute error by almost 1 dollar on train set and a lower MAE on test set by more than 2 dollars. It also exhibits less variability . therefore we will use this model to predict "Big mountain resort's" ticket price.

Take aways during – Modeling phase

Method details:

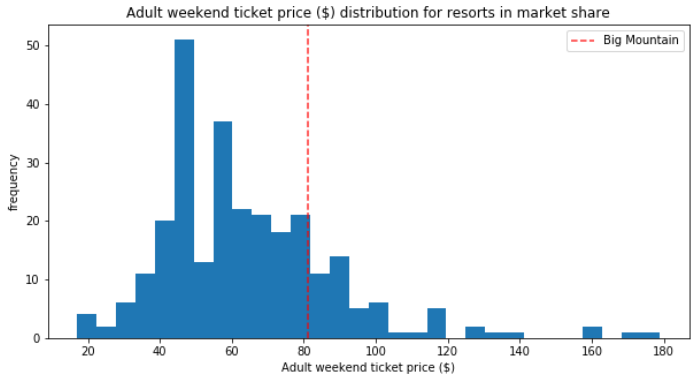
1. Load best model and load data
2. Refit model on all available data (excluding Big mountain)

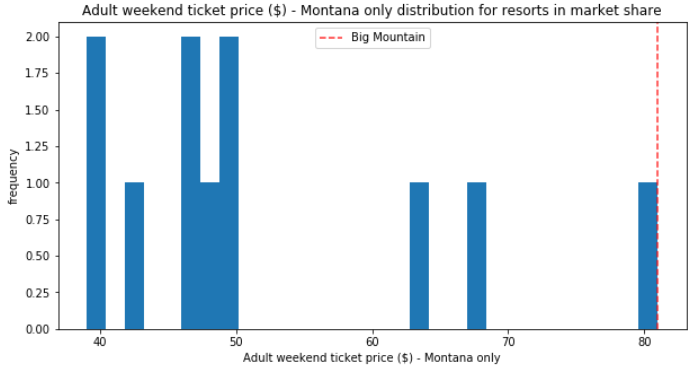
This next step requires some careful thought. We want to refit the model using all available data. But should we include Big Mountain data? On the one hand, we are not trying to estimate model performance on a previously unseen data sample, so theoretically including Big Mountain data should be fine. 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 **solely** on its competitors.

2.1 fit mode - on all 276 records

2.2 cross validate – to recieve wanted scoring (MAE/R^2..)

where 'Big Mountain' sits overall amongst all resort for price





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.

SUMMARY OF MODELING PHASE

After refitting the model on the entire data excluding Big Mountain resort, MAE is: 10.385 with 1.487 std. only slightly higher compared to MAE on the train set alone 9.659 with 1.349 std.

Currently Big Mountain charges 81 dollar, and although state-wise Big Mountain's ticket price is the highest and sits high amongst all resorts, there are still resorts with a higher price and up to double of the price.

Our modelling suggests that charging 94.22 dollar per ticket could be fairly supported in the marketplace by Big Mountain's facilities.
Features that came up as most valued by costumers in the modeling include:
1. FastQuads - Most resorts have no fast quads. Big Mountain has 3
2. Runs - Big Mountain compares well for the number of runs
3. Snow Making area - our resort is very high up the league table of it
4. Vertical drop - wich our resort is doing well, and there are few with greater drop
5. Big Mountain is amongst the resorts with the largest amount of skiable terrain.
6. Total number of chairs - our resort has amongst the highest number of total chairs

After seeing where Big mountain resort stands amogst all in those ereas, we feel confident that existing resorts' facilities can support the higher ticket price.

Big Mountain Resort has recently installed an additional chair lift to help increase the distribution of visitors across the mountain. This additional chair increases operating costs by 1.54 Million dollar this season. On the basis of each visitor on average buying 5-day tickets and each year ther are about 350,000 visitors at the resort, only to cover this investment, price should be raised in 0.88 dollar per ticket.

For further improvements, we would recommend the following:
First, to permanently close down the one least used run, as it has shown from the model that it is not reducing ticket value.
Then, our model shows that increasing the vertical drop in 150 feet by adding additional lower run (second scenario) supports a 1.99 dollar increase in ticket price. Over the season, this could be expected to amount to 3.475 Million dollar.
Implimentation of this scenario requires the installation of an additional chair lift to bring skiers back up, Resort's previouse experience shows that price should be raised in 0.88 dollar per ticket to cover additional chait lift installation. It is an extra revenue of aproximatly 1.935 Million dollar in the first season and 3.474 Million dollar for each following year.
Only after knowing additional chair lift's yearly operatinal costs, we could calculate the profit.
It's worth to mention that addind 2 acres of snow making coverage make no difference on ticket value therfore does not worth the investment.

Lastly, resort should look up at its runs' operational costs and make decision when should it close permanently close these least used runs:

- If Yearly operational costs of 2 runs greater 675K dollars, recomendation to Permanently close down 2 least used runs
- If Yearly operational costs of 5 runs greater 1.225 M dollars, recomendation to Permanently close 5 least used runs
- If Yearly operational costs of 8 runs greater 2.2 M dollars, recomendation to Permanently close down 8 least used runs. As we can see in the graph in 1st scenario section, 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, so as 6,7 and 8.