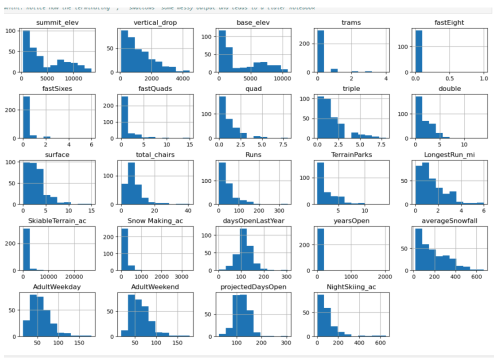
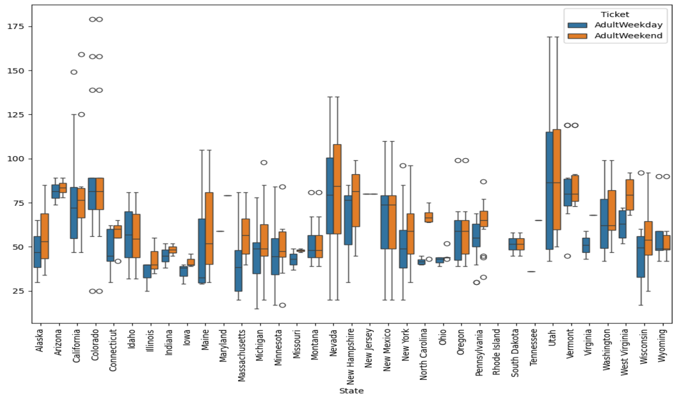
Big Mountain Final Report

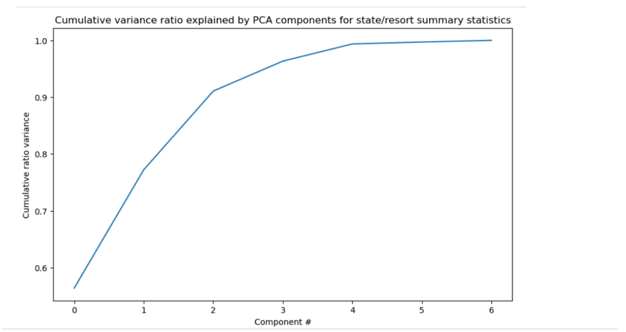
The Big Mountain Resort is a skiing resort in Montana. It offers great views of Glacier National Park and Flathead National Forest. They offer access to 105 trails, 11 lifts, 2 T-bars, and 1 magic carpet for novice skiers. The longest run is named Hellre and is 3.3 miles in length. The base elevation is 4,464 ft, and the summit is 6,817 ft with a vertical drop of 2,353 ft. The resort has recently installed an additional chair lift to help increase the distribution of visitors across the mountain. This additional chair increases their operating costs by $1,540,000 this season. There are 350,000 yearly visitors. The Big Mountain resort wants to know What can they do to capitalize on some of its facilities that are more important than others facilities, so that they can cut costs without undermining the ticket price or can support an even higher ticket price by the next season?

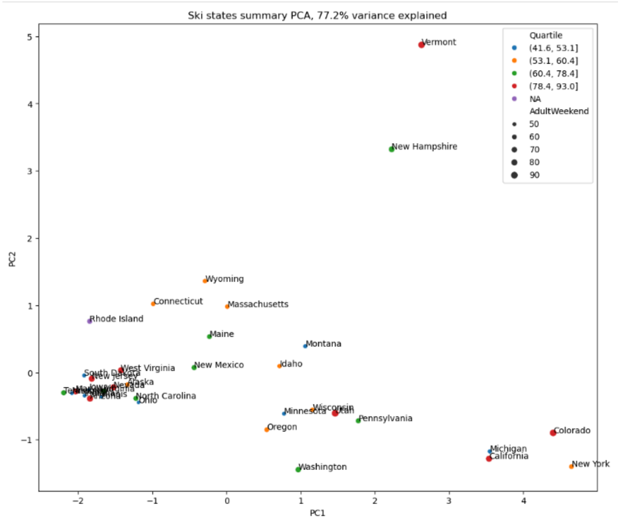
Furthermore, with the available ski resort dataset of all the resorts in the US we performed data wrangling. First we found out which columns have the most null values which is fastEight. We also found AdultWeekday and AdultWeekend prices have quite some missing values. We also learned that all rows have unique data points. Then we plotted all the features to see the distribution of values. We further analyzed and found that for Silverton Mountain the value of SkiableTerrain_ac is wrong so we changed it by researching on the web. We dropped the fastEight column because it has 163 missing values. We learned that in yearsOpen field one value is 2019 which is not possible so we dropped that row. This could be because instead of stating how year it is opened they put the year it was opened. We dropped the rows which had null values for both AdultWeekday and AdultWeekend fields. We also dropped the column AdultWeekday because it has more missing values than AdultWeekend and we drop rows that are missing values for AdultWeekend. It is important to normalize the data with US states Population and area data so we can understand resort to population and resort to area ratios. Then we merge these columns to the state_summary dataframe which has resorts_per_state, state_total_skiable_area_ac, state_total_days_open, state_total_terrain_parks, state_total_nightskiing_ac columns. Lastly we save the two datasets ski_date and state_summary using save_file function.

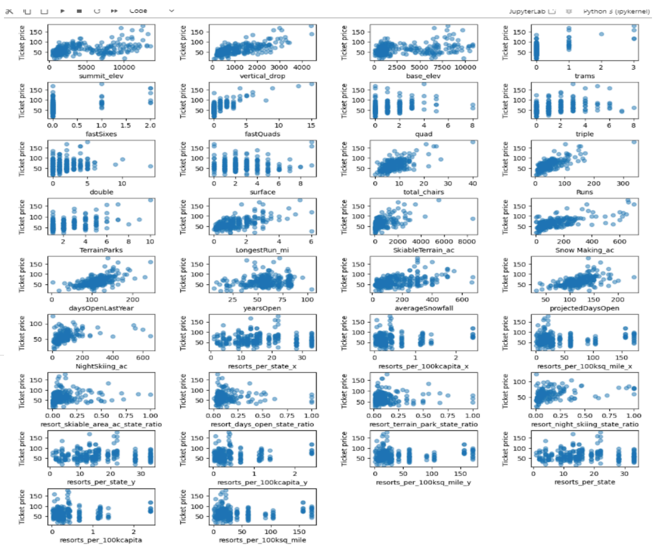




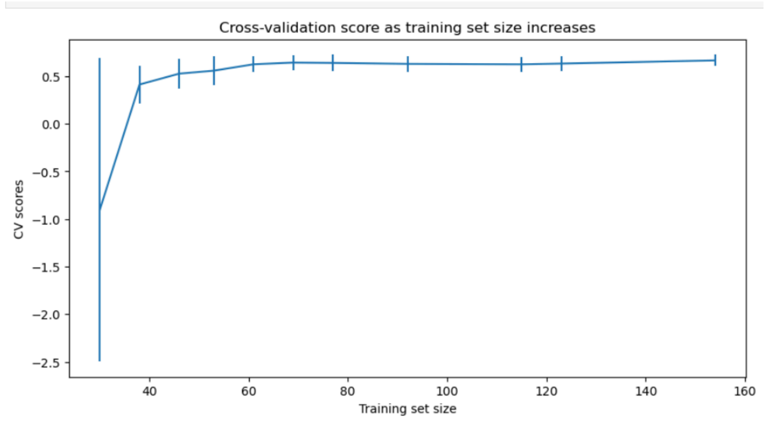
Moreover, after cleaning the data we want to do Exploratory Data Analysis(EDA) which helps show the relationship between features and find hidden details about the data that we did not know. We wanted to know how many resorts per population and how many resorts per state area so we can understand how it relates to pricing. We add resorts_per_100kcapita and resorts_per_100ksq_mile columns in state_summary. Then we scale all the numeric data points on every field so we get a common scale. To gather further analysis we calculated PCA over the dateframe. The PCA tells us which components rank high on importance and its impact on the data. It shows that two components account for 75% of variance. We plotted the two component's PCA, PC1 vs PC2 with AdultWeekend pricing as range labels. Then we merged state_summary to ski_data to do feature engineering. We added these columns resort_skiable_area_ac_state_ratio, resort_days_open_state_ratio, resort_terrain_park_state_ratio , and resort_night_skiing_state_ratio while we removed the state related columns. We created a heatmap of all the features to visualize the correlation between all the features. Furthermore, we wanted to see the relationship between the price and all the features, so we created a function called scatterplots. We notice vertical_drop, Runs, total_chairs have strong correlation with pricing. From that we created more columns to understand total_chairs over Runs, total_chairs over SkiableTerrain, fastQuads over Runs, and fastQuads over SkiableTerrain.

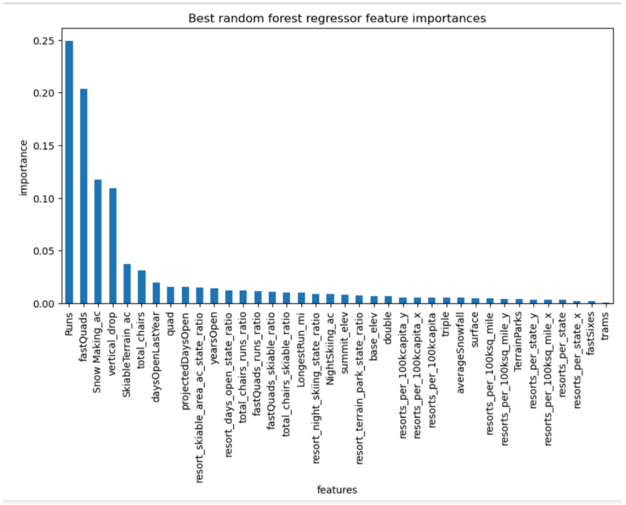






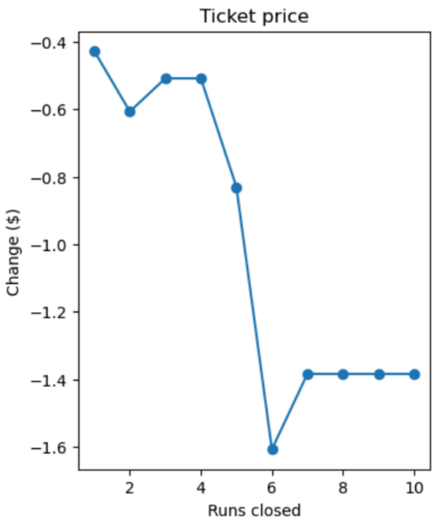
Next, we did preprocessing to select and train our model so it can predict pricing. We have to split the data into training and testing. We dropped the cloumans which had an object as type because we are only focused on numeric values and wand to predict a numeric value. We want to test a model so we picked DummyRegressor and used the mean to get the final result. We calculated r_squared error, mean squared error and mean absolute error to see how well the model is doing train vs test. We want to train our data in the proper way so we get the best result possible. So, we know from past analysis that there are missing values in our dataset so we imput the date with the median values. Then we scale the date so all the values are normalized. We use a linear regression model to train on the data and use test data to predict. We use r_squared error, mean squared error and mean absolute error to see the performance of our data. For a different approach we can imput the missing values with mean and do all the steps after but we found the result was not significant. Likewise there are four steps: impute missing values, scale the features, train a model, calculate model performance so we can create a pipeline to simplify the process. So, we use the function make_pipeline to create a pipe which helps us see all the important steps in one place. Our goal was to understand what features have more importance on the pricing so, we added a SelectKBest function in the pipeline to see what and how many features are needed. We tried some values for K but it did not help narrow down. We found with lr_grid_cv.best_params_ method that there are eight features that are the most important. They are vertical_drop, Snow Making_ac, total_chairs, fastQuads, Runs, LongestRun_mi, trams, SkiableTerrain_ac. Another question came up which is how the model will do with multiple iterations so we use cross validation to check. We wanted to check a different model to see what the results are, so we are biased towards one model if others models are better. So, we tried a random forest model and it performed better. At the end we picked the random forest model and saved the model to our folder.

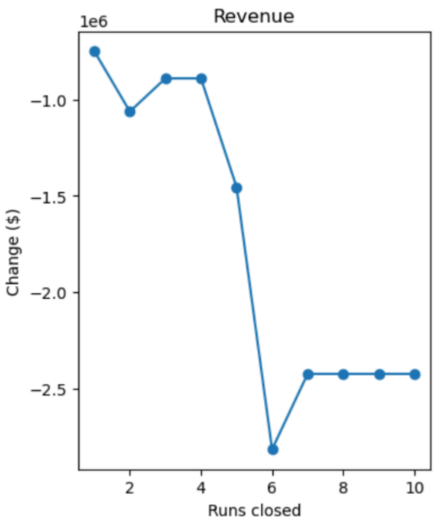




Final step is modeling which to predict pricing for our Big Mountain resort based on our model and see what types of options we have to solve our problem. So, we train our model again but without the Big Mountain resort then we predict the pricing with just Big Mountain to get our result. The output *Big Mountain Resort modelled price is $89.45, actual price is $81.00. Even with the expected mean absolute error of $10.29,this suggests there is room for an increase.* We wanted to see where Big Mountain lies with other resorts in the eight features that are important and we found Big Mountain is above average in most of them. After that we came up with four scenarios that help us understand more about pricing if we change some features like Runs and total_chairs. In the first scenario we closed down 10 runs permanently and found if we close 4-6 runs the pricing drops while after closing 6 the pricing has no change. In the second scenario we added a run, increased the vertical drop by 150 feet, and installed an additional chair lift. The result increases support for ticket price by $0.44. In the third scenario we are repeating the previous one but adding 2 acres of snow making and the result is similar to the last one with support of $0.44 increase. In the last scenario we increase the longest run by 0.2 mile, requiring an additional snow making coverage of 4 acres the result no difference.

```
Pipeline(steps=[('simpleimputer', SimpleImputer(strategy='median')),
                ('standardscaler', None),
                ('randomforestregressor',
                 RandomForestRegressor(n_estimators=112, random_state=4))])
```


Ticket price
Change ($)
Runs closed
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
−1.6
2
4
6
8
10


Revenue
1e6
Change ($)
Runs closed
−1.0
−1.5
−2.0
−2.5
2
4
6
8
10