**Documentation**

Introduction

Big Mountain Resort is a ski resort located in Montana. Every year about 350,000 people ski or snowboard at Big Mountain. Big Mountain 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. The resort's pricing strategy has been to charge a premium above the average price of resorts in its market segment. The business wants some guidance on how to select a better value for their ticket price to meet the increased operating costs.

Problem statement

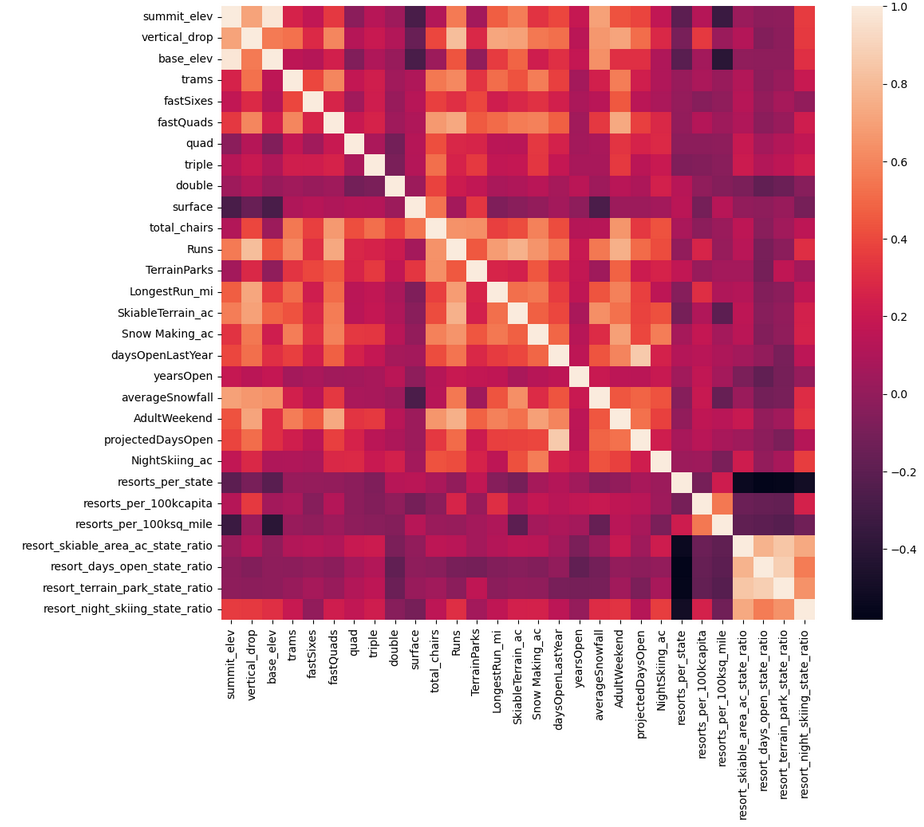
Big Mountain Resort has to select a better value for their ticket price for the upcoming season to meet the increased operating cost of $1,540,000; so that their revenue is increased.

Data Wrangling

This step focuses on collecting your data, organizing it, and making sure it's well defined. There were total 330 entries and 27 columns present on the Ski resort data at the beginning. Our resort 'Big Mountain Resort' details was also included. Our resort doesn't appear to have any missing values. Based on the information we collected, we identity certain wrong values and we corrected those (Silverton Mountain Resort's skiable area). We have no ticket pricing information at all for Heavenly Mountain Resort, so we will simply be dropping the entire row! Drop the fastEight column in its entirety; half the values are missing and all but the others are the value zero. We can certainly state that no resort will have been open for 2019 years! It likely means the resort opened in 2019. It could also mean the resort is due to open in 2019. You don't know when these data were gathered! So simply drop this row. By using a fairly new groupby behaviour called named aggregation, statewide summary was created. As state-wide supply and demand of certain skiing resources may well factor into pricing strategies. \*TerrainParks \*SkiableTerrain\_ac \*daysOpenLastYear and \*NightSkiing\_ac. About 14% of the rows have no price data. As the price is your target, these rows are of no use. So those rows are removed. After removing those rows, distributions are much better. Population and area data for the US states can be obtained from wikipedia. This table of data is useful because it allows you to easily pull and incorporate an external data set. It also allows you to proceed with an analysis that includes state sizes and populations for your 'first cut' model. When modeling the ticket price, a couple of observations can be made. Firstly, there is a clear line where weekend and weekday prices are equal. 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. There are still some missing values in the data, we don't know is how useful the missing features are in predicting ticket price. So we shouldn't just drop rows that are missing several useless features. So, at last 277 rows and 25 columns were left in the data.

Exploratory data Analysis

This step focuses on summarizing and visualizing data and find out patterns and make hypothesis. At the beginning of exploratory data analysis, there were total 27 columns with 3 non-numeric columns. We first explore data from state\_summary. After calculating resort density, it is okay to drop the absolute population and state size columns from state\_summary and add the resort density columns. So we now got a Dataframe that speaks to the skiing competitive landscape of each state. After calculating PCA, we find out that the first two components seem to account for over 75% of the variance, and the first four for over 95%. When visualizing Ski states summary PCA, 77.2% variance was explained. It was done both using matplotlib and using seaborn. 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 is Feature correlation heatmap. Many analysis can be made from it.



Correlations, particularly viewing them together as a heatmap, can be a great first pass at identifying patterns. But correlation can mask relationships between two variables. So we create a series of scatterplots to really dive into how ticket price varies with other numeric features. We then calculate and add total\_chairs\_runs\_ratio, total\_chairs\_skiable\_ratio, fastQuads\_runs\_ratio, fastQuads\_skiable\_ratio to our ski\_data; It seems logical that this ratio would inform you how easily, and so quickly, people could get to their next ski slope! After completing EDA, we have 3 non-numeric column and 33 numeric columns in ski\_data.

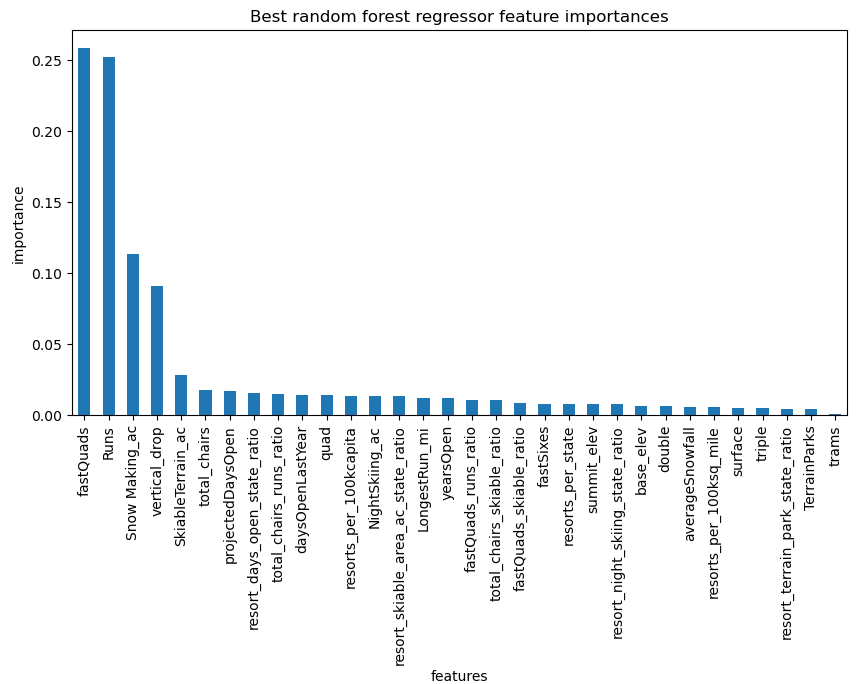
Model Preprocessing with feature engineering

For preprocessing, we separate the Big Mountain data from the rest to use later. We consider ski resort data as a single entity. 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. An often overlooked subtlety here is that people all too frequently use the test set to assess model performance and then compare multiple models to pick the best. This means their overall model selection process is fitting to one specific data set, now the test split. You could keep going, trying to get better and better performance on that one data set, but that's where cross-validation becomes especially useful. While training models, a test split is very useful as a final check on expected future performance. So we divide our data as 70/30train/test split.

Algorithms used to build the model with evaluation metric

There are many ways of assessing how good one set of values agrees with another, which brings us to the subject of metrics. We consider R^2; coefficient of determination -as the metrics and calculate it for different training and testing sets. But R^2 is less appealing if you want an idea of how "close" your predictions are to the true values. Metrics that summarise the difference between predicted and actual values are mean absolute error and mean squared error. So we calculate it for different training and testing sets. With sklearn.metrics (provides many commonly used metrics), we can calculate the above values easily without using functions. We then impute missing values using the median, apply the imputation to both train and test splits, scale the data, train the model on the train split, make predictions using the model on both train and test splits and assess model performance. Instead of using functions to perform all these, we use sklearn's pipeline. A single call to the pipeline's fit() method combines the steps of learning the imputation, the scaling, and then training the model.

sklearn has a number of feature selection functions available. The one we use here is SelectKBest which selects the k best features. f\_regression is just the score function you're using because you're performing regression. Next we started cross-validation. It means 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. We are now using a built in function in sklearn. This is GridSearchCV. By using this, we could write a for loop and iterate over each possible value, doing all the housekeeping ourselves to track the best value of k. A model that can work very well in a lot of cases is the random forest. For regression, this is provided by sklearn's RandomForestRegressor class.It stop the bad practice of repeatedly checking performance on the test split. Instead, go straight from defining the pipeline to assessing performance using cross-validation. After performing Random Forest Regression and plotting the best features; the dominant top four features are in common with your linear model: • fastQuads • Runs • Snow Making\_ac • vertical\_drop(plotted below). 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. To assess the data quality, we plot a learning curve. This shows that we 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.



Winning model and scenario modelling

We calculate expected Big Mountain ticket price from the model. Big Mountain Resort modelled price is $95.87, actual price is $81.00. Even with the expected mean absolute error of $10.39, this suggests there is room for an increase. This result should be looked at optimistically and doubtfully! The validity of our model lies in the assumption that other resorts accurately set their prices according to what the market (the ticket-buying public) supports. The fact that our resort seems to be charging that much less that what's predicted suggests our resort might be undercharging. But if ours is mispricing itself, are others? It's reasonable to expect that some resorts will be "overpriced" and some "underpriced." Or if resorts are pretty good at pricing strategies, it could be that our model is simply lacking some key data? Certainly we know nothing about operating costs, for example, and they would surely help.

Big Mountain Resort has been reviewing potential scenarios for either cutting costs or increasing revenue (from ticket prices).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.

• 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.

• In this scenario, Big Mountain is adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift; which increases support for ticket price by $1.99. Over the season, this could be expected to amount to $3474638.

• In this scenario, you are repeating the previous one but adding 2 acres of snow making. This scenario increases support for ticket price by $1.99. Over the season, this could be expected to amount to $3474638. Such a small increase in the snow making area makes no difference!

• This scenario calls for increasing the longest run by .2 miles and guaranteeing its snow coverage by adding 4 acres of snow making capability. No difference whatsoever.

Pricing recommendation

The modeled scenarios we'd recommend for further consideration is Big Mountain adding a run, increasing the vertical drop by 150 feet, and installing an additional chair lift; which increases support for ticket price by $1.99(So the new ticket price is $82.99). Over the season, this could be expected to amount to $3474638.

Conclusion

We are creating our model to gain insight into what Big Mountain's ideal ticket price could/should be, and how that might change under various scenarios. Currently the adultweekend price for Big Mountain Resort is $81. We want to refit the model using all available data. The motivation for this entire project is based on the sense that Big Mountain needs to adjust its pricing. One way to phrase this problem: we want to train a model to predict Big Mountain's ticket price based on data from all the other resorts! We don't want Big Mountain's current price to bias this. We want to calculate a price based only on its competitors.

Future scope of work

The criterion for success is increased revenue for the resort on the upcoming season. The scope of solution space is to either cut their costs without undermining the ticket price or support an higher ticket price. If business wanted to test a new combination of parameters in a scenario in future, they can refer the documentation provided by us with details of how the change in different features affect their ticket price in different scenario.