The Problem

Big Mountain is a ski resort with 105 trails, 11 lifts, 2 T-bars and a magic carpet. They recently installed an additional chair lift to help increase the distribution of visitors across the mountain. 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 to get a better value for their ticket prices. They want to either cut costs without undermining ticket prices or support a higher ticket price. What possibilities does Big Mountain have to raise its prices this season in order to offset the $1,540,000 investment for an additional chair lift?

Recommendations

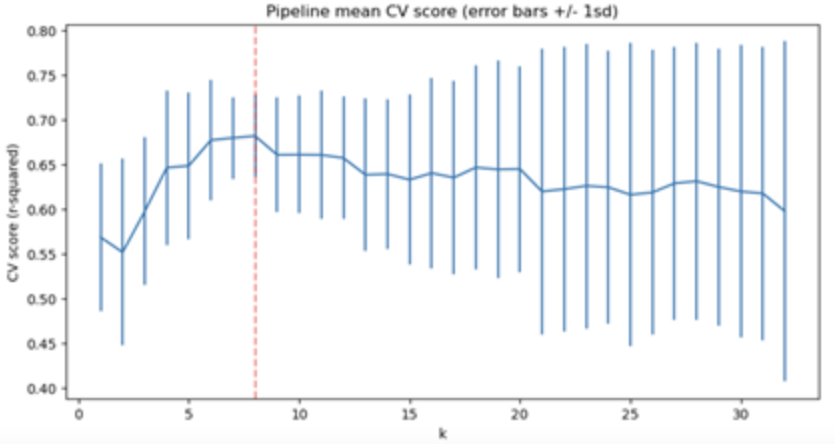
Currently Big Mountain is charging $81 per weekend/weeknight. Using our Random Forest Regressor model the price Big Mountain should be charging is $95.87. Using all of the analysis with potential changes to Big Mountain the options that made the most sense and had the greatest impact on ticket revenue were adding an additional run, increasing the vertical drop by 150 ft and adding an additional chair lift. In order to see further into this option, we would need to know the operating cost of a new chair lift per ticket in the context of raising prices to cover this. Another potential to change revenue is for Big Mountain to close up to 5 runs with less than $0.75 loss per ticket on sales.

Model Training:

First I used the average of ticket prices as a baseline to test several models against. Then I built several different learning models. There were some test models such as the SimpleImputer that had different levels of accuracy based on different K values as shown below:

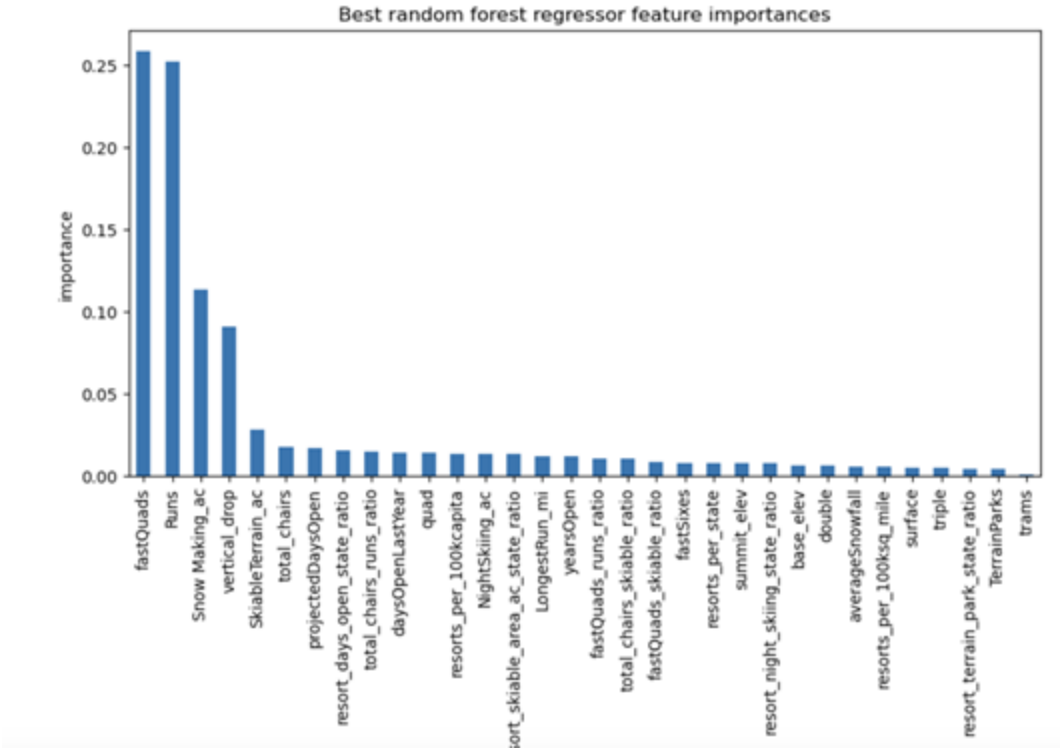
```
[164]: lr_grid_cv.fit(X_train, y_train)

[164]: GridSearchCV
       estimator: Pipeline
         SimpleImputer
         StandardScaler
         SelectKBest
         LinearRegression

[166]: score_mean = lr_grid_cv.cv_results_['mean_test_score']
       score_std = lr_grid_cv.cv_results_['std_test_score']
       cv_k = [k for k in lr_grid_cv.cv_results_['param_selectkbest__k']]

[168]: #Code task 19#
       #Print the `best_params_` attribute of `lr_grid_cv`
       lr_grid_cv.best_params_

[168]: {'selectkbest__k': 8}
```

```
[170]: #Code task 20#
       #Assign the value of k from the above dict of `best_params_` and assign it to `best_k`
       best_k = lr_grid_cv.best_params_['selectkbest__k']
       plt.subplots(figsize=(10, 5))
       plt.errorbar(cv_k, score_mean, yerr=score_std)
       plt.axvline(x=best_k, c='r', ls='--', alpha=.5)
       plt.xlabel('k')
       plt.ylabel('CV score (r-squared)')
       plt.title('Pipeline mean CV score (error bars +/- 1sd)');
```

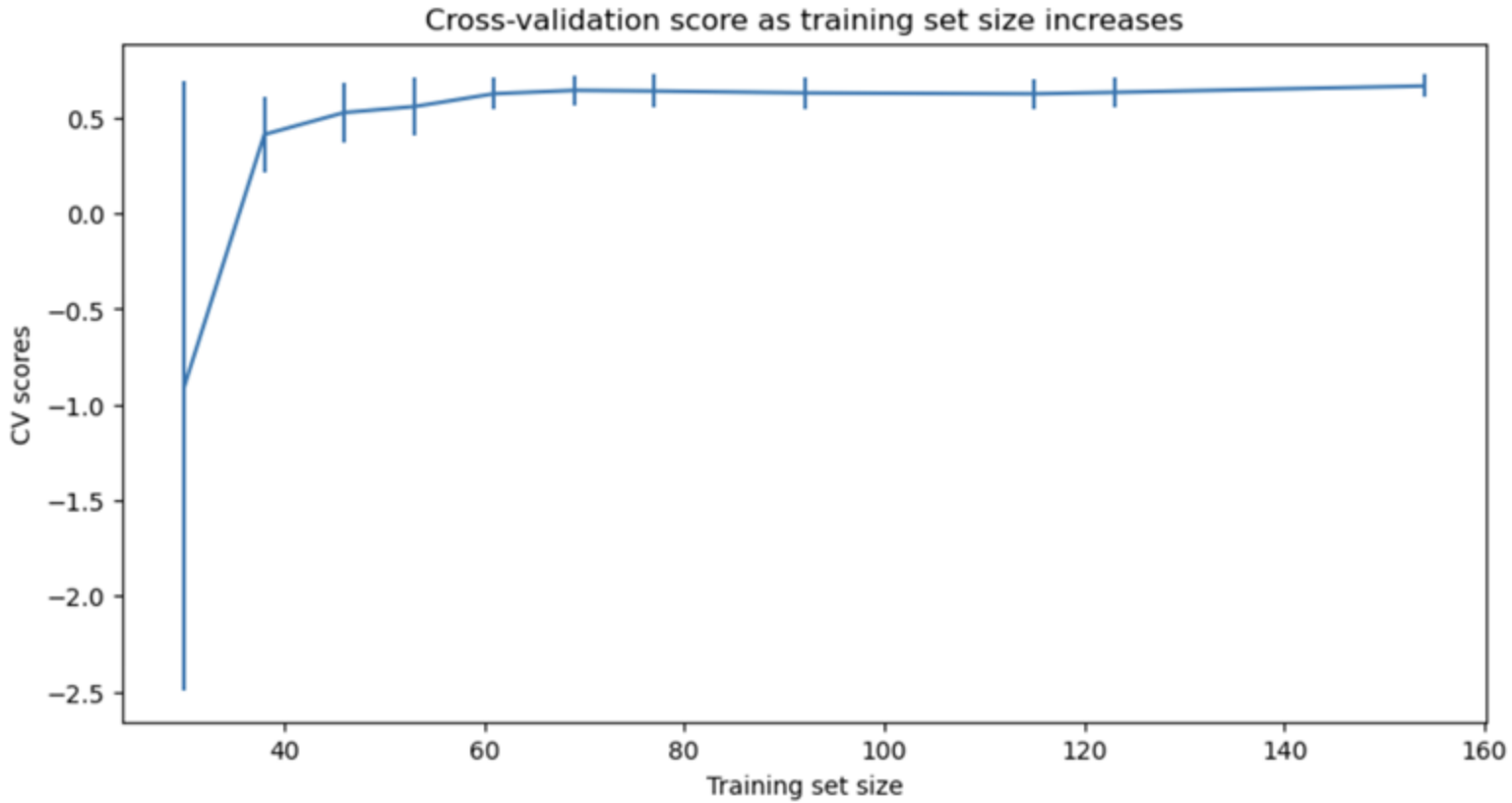


The above suggests a good value for k is 8. There was an initial rapid increase with k, followed by a slow decline. Also noticeable is the variance of the results greatly increase above k=8. As you increasingly overfit, expect greater swings in performance as different points move in and out of the train/test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

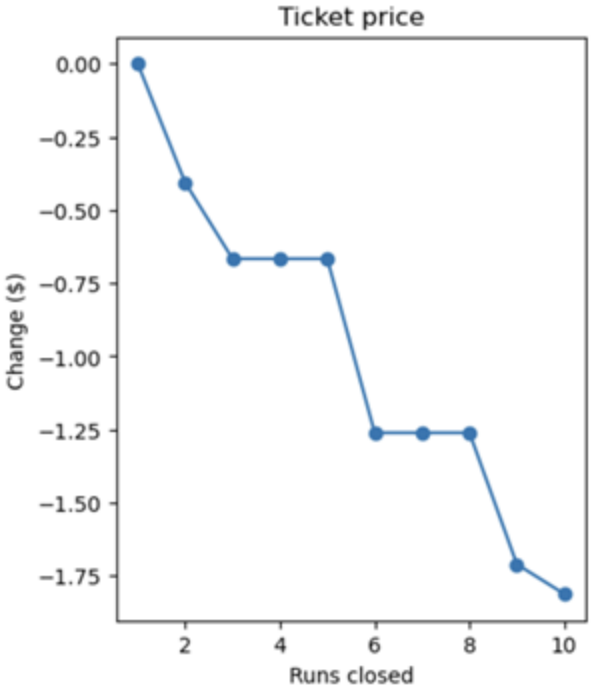
I could have kept trying to fit the data to different models by tweaking the values used, but ultimately the random forest regressor worked best because it had a lower cross validation mean absolute error and highlighted the same key features that we determined in our linear model as shown below:

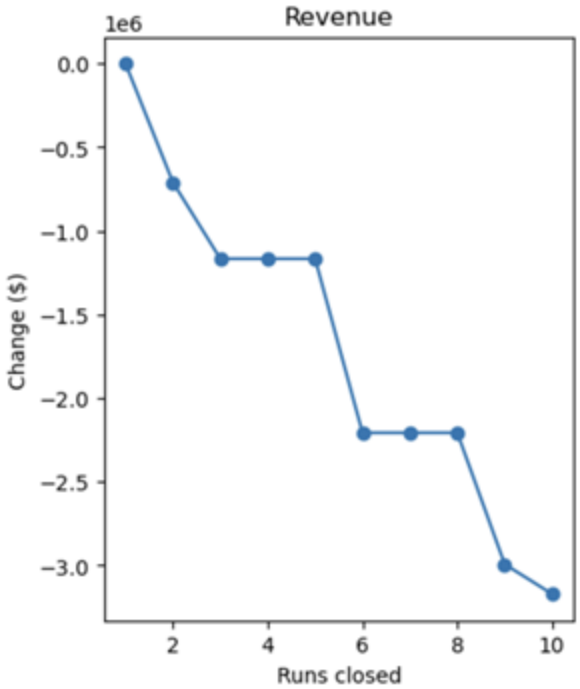


This shows that you seem to have plenty of data. There's an initial rapid improvement in model scores as one would expect, but it's essentially leveled off by around a sample size of 40-50.



The model we used 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.





When we used our model to test if there would be any change by adding 2 acres of snow making land or increasing the longest run mileage by 0.2 miles, it made no difference in the predicted revenue from our model.

[67]:
```
#Code task 4#
#Call `predict_increase` with a list of the features 'Runs', 'vertical_drop', and 'total_chairs'
#and associated deltas of 1, 150, and 1
ticket2_increase = predict_increase(['Runs', 'vertical_drop', 'total_chairs'], [1, 150, 1])
revenue2_increase = 5 * expected_visitors * ticket2_increase
```

[69]:
```
print(f'This scenario increases support for ticket price by ${ticket2_increase:.2f}')
print(f'Over the season, this could be expected to amount to ${revenue2_increase:.0f}')
```

```
This scenario increases support for ticket price by $1.99
Over the season, this could be expected to amount to $3474638
```

Conclusion:

I want to see how business executives feel about this model and see if they'd like to increase their ticket prices to something more reflective of the model (closer to $95.87) in addition to making the changes mentioned with the additional run, increase in vertical drop by 150 ft and additional chair lift being added. Business analysts could use the function 'predict_increase' on different parameters to see what would be most effective in supporting continued growth at Big Mountain in the future.