Prepared by:
Aaron Burdette

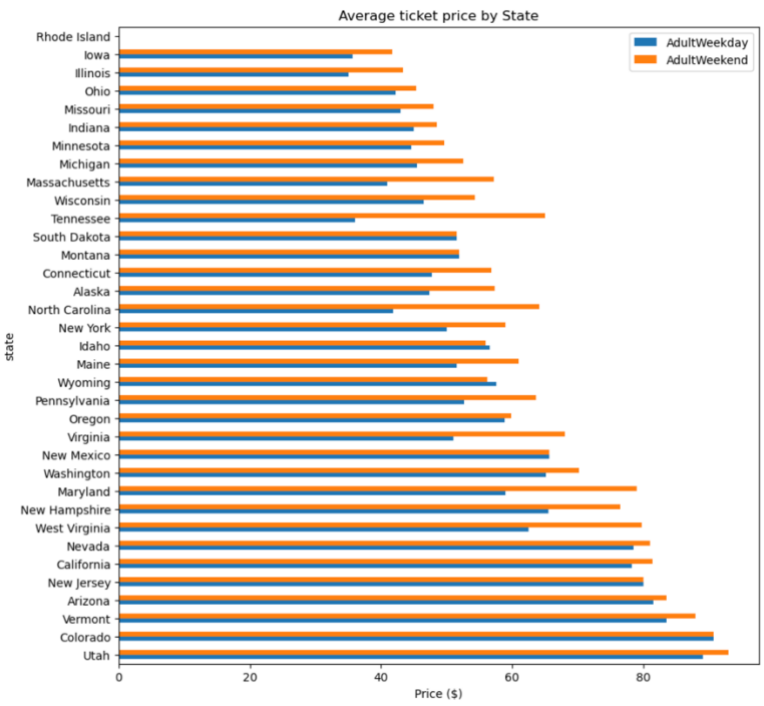
Guided Capstone Project Report

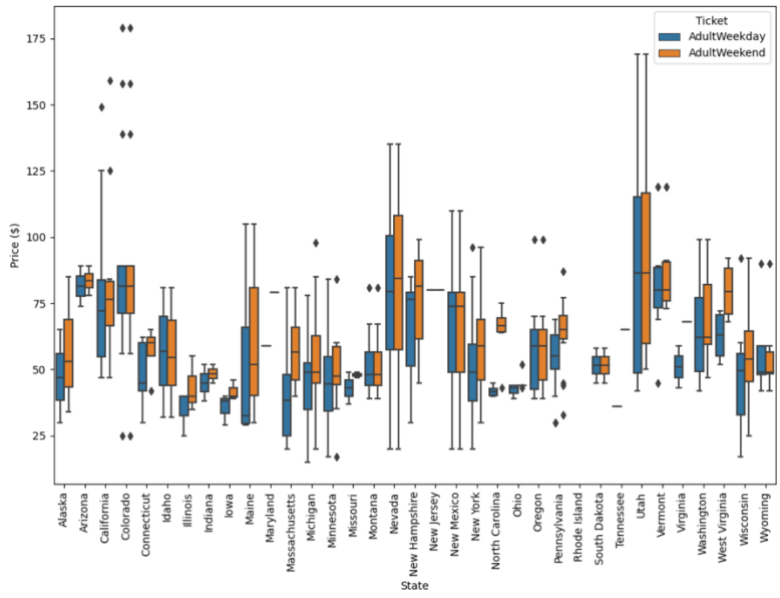
Objective:

Creating a model to predict ski resort ticket pricing and use it to study Big Mountain's current and potential future pricing strategy. The aim is to see if BM Resort's current ticket price makes sense relative to its peers; additionally, the model aims to study how the addition/removal of certain resort facilities (features) would affect changes in ticket pricing.

Methodology:

Ski data was acquired using the .csv file provided and loaded into a data frame (df). The ski_data df was then checked for duplicate entries. The mean ticket price for each state was calculated then assigned to state_price_means df for further exploration as shown below. Furthermore, each state's ticket price ranges were depicted using boxplots:

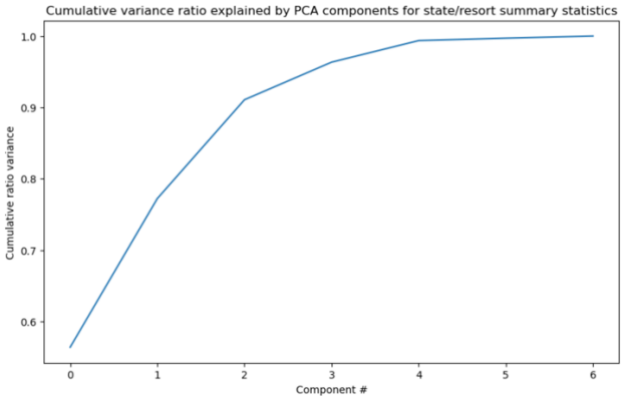




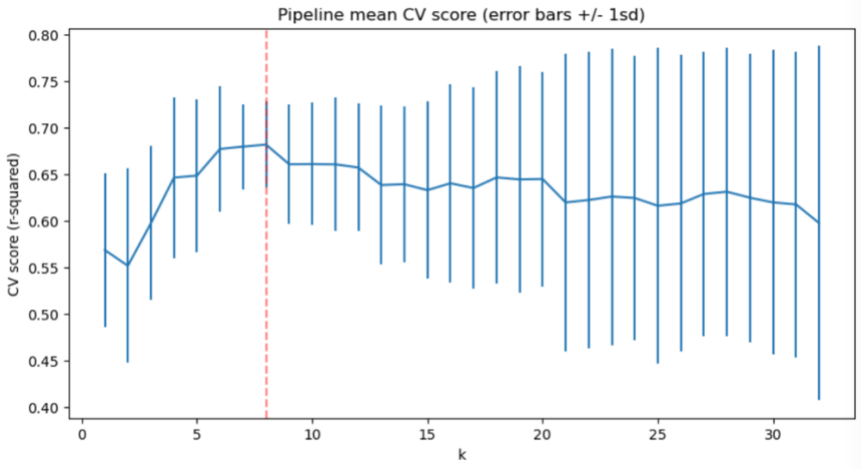
In the data wrangling step ticket prices were also dropped. Specifically, resorts with only one price were kept as they contained some useful information; resorts with both Weekend and Weekday prices missing, however, were dropped from ski_data since I would not have any price data to work from. Additionally, each features' histograms were created then examined. Features that had distributions that were non-normal or had abnormal outliers were more closely examined. Those outliers were either corrected to the correct values or dropped if they were unverifiable. Population & land area data was imported from Wikipedia and stored in the usa_states df. State population and state area (in sq_mi) data were then added to the original ski_data for use in later steps.

During the EDA phase, the 2 new features created were 'resorts_per_100kcapita' and 'resorts_per_100ksq_mile' by taking 'resorts_per_state' and dividing it by 'state_population' and 'state_area_sq_miles' respectively. When performing PCA on scaled state_summary data, it was shown that that first 3 components accounted for 90%+ of the variance in the data:

Prepared by:
Aaron Burdette



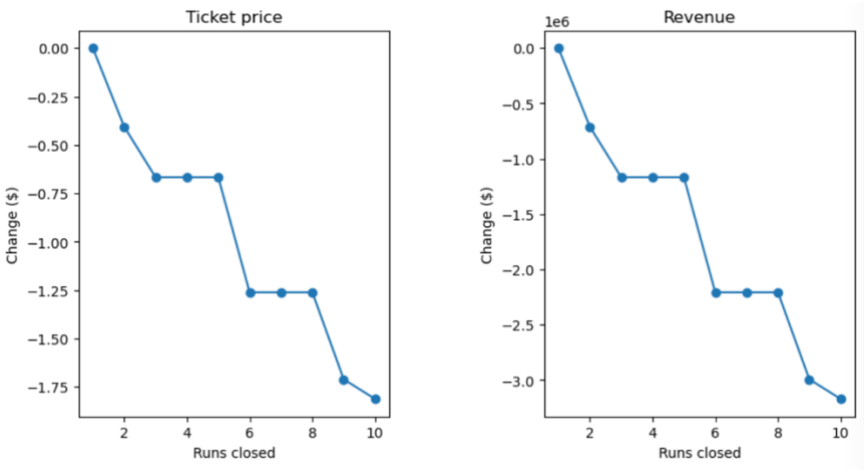
For the Pre-processing and Training phase of this project, Big Mountain resort was first removed from ski_data to. Ski_data's entries were then split into 70% training data while 30% were reserved for testing. The reserved testing data can be used to evaluate the predicted target feature by calculating metrics such as R2, MAE, and MSE. Categorical features like Name, state, and Region were dropped prior to modeling. Before modeling even began, many features had missing values. These missing values could be imputed in many ways, but for this model, the mean of the features was used to fill in the missing feature data. First, a simple Linear model pipeline was made utilizing make_pipeline(). Its arguments consisted of SimpleImputer(strategy='median') which imputed missing X values with feature medians, X values were then scaled using StandardScaler(), and finally the data would be fitted to a linear model using LinearRegression(). To use the simple LR model pipeline, I called the .fit(X_train, y_train) method on the Pipeline object and fitted the training data. After which, calling the .predict() method on the Pipeline object while using either X_train or X_test as arguments would yield their predicted y values respectively. The simple LR model pipeline was refined further by adding SelectKBest(f_regression) which choose the features that yielded the best model. This step is added to ensure that data would not be overfitted to the model. Cross-validation was then performed on the LR model pipeline which essentially splits the training data into a specified number of 'k-folds' and reserves one fold for testing. CV does this for 'k' times as it iterates thru each fold and returns test_scores for each fold of the training data. Cross-validation can also be performed by using sklearn's GridSearchCV() and passing in the pipeline & grid_params for SelectKBest. After fitting X_train and y_train data on the GridSearchCV(), the test results can then be accessed via GridSearchCV()'s .cv_results_ method. With the answer of the best number of k-folds to use, I could then refine the model with the k value:



A RF_pipe was also created using make_pipeline(). It consists of SimpleImputer(strategy='median'), StandardScaler(), and RandomForestRegressor(random_state=47). Hyperparmeter search using GridSearchCV() was also performed on the RF_pipe. The best parameters for the RF_pipe was the following: {'randomforestregressor__n_estimators': 69, 'simpleimputer__strategy': 'median', 'standardscaler': None}. As

the Random Forest regression model had a smaller MAE, it would be better suited to use it in predicting ticket prices.

Lastly in the modeling phase of the project, the Random Forest regression model created in the training step would finally be used to model Big Mountain Resort's ticket price. Before predicting BM Resort's price, it was first dropped from the ski_data df. The model was then fitted to the entirety of the ski_data. A df (X_bm) containing only BM Resort relevant features for modeling was created and then used to predict its price (bm_pred). The model predicts that BM Resort's fair ticket price is $95.87 whereas it currently charges $81. Other potential prices were then modeled with different assumptions in the changes of features, i.e., how would adding/removing certain resort facilities affect BM Resort's ticket price. The chart below showcases the change in ticket prices as runs are closed:



Analysis/Executive Recommendation:

It is recommended to bring the current AdultWeekend price of BM Resort in line with the modeled prediction or consider adding/removing certain resort facilities (features) and see how the modeled price responds to said alterations. Additionally, current provided data can only model top-line revenues; with the addition of cost information, more insights on bottom-line profit(loss) can be further explored.