Slide 1: Greetings everyone, we are group 8 and this is our SC1015 mini project, titled Bike Sharing. My team consists of Joanthan Chow, Khant Zaw and myself Jacob Rossman

Slide 2: The real life problem we identified is to help bike sharing companies improve user experience for riders. The data science problem that we have extracted is a Response prediction problem. (Are we able to forecast use of a bikeshare system? And Which model would be the best to predict?).

Slide 3: Here are the steps we took.

Slide 4: To solve this problem, we used a publicly available bike sharing dataset from kaggle. The size of this dataset was initially 10887. As you can see here, our dataset contains various data types, including boolean, floats, integers, and datetime.

Slide 5: Here are the descriptions of the different features. 'Count' here will be our response variable, which is the total rentals in each hour. We will disregard 'casual' and 'registered' as the sum of these 2 is our 'count'. The other variables will be our predictors. For instance, season is a categorical variable listing the 4 seasons spring, summer, fall and winter. Holiday and workingday are boolean variables. Weather is a categorical variable that differentiates how extreme the weather is. Atemp is the "feels like" temperature in celsius. The remaining variables temp, humidity, and windspeed are self-explanatory numerical values. So after the preliminary data exploration of our dataset, we have narrowed down our useful data to these fields.

Slide 6: Moving on to cleaning the dataset. Upon checking the description of our dataset, we found that all the variables were listed as 'objects', so we converted the variable data into suitable data types, and the end result is shown here.

Slide 7: Notice here that although 'season', 'holiday' , 'workingday' and 'weather' are categorical variables, we have converted them to numeric, as we intend to compute their correlations using Point Biserial Correlation Coefficient (PBCC) and Phi Coefficient, which will be explained later. We also checked for null values and outliers and removed them, reducing the size of our dataset to 10586.

Slide 8: Next we performed exploratory data analysis to understand the distribution of our variables and any potential relationships between them. First, this is how our response variable 'count' is distributed. We can see that distribution is positively skewed, meaning that there are more rentals on the lower end of the scale and fewer rentals on the higher end. This suggests that there may be a long tail to the right in the distribution of the count variable, which may need to be addressed during data preprocessing or modeling.

Slide 9: Next, we used barplots to visualize how our categorical variables vary with count, and we used histograms with KDE to visualize how our numerical variables vary with count.

Slide 10: To analyze our numerical variables, we used a correlation matrix. From here, although the correlation between them and count is not very prominent, we can see that count has some dependency on temp and humidity. Additionally, we will omit atemp, as atemp and temp are highly correlated, and using it might cause overfitting later on.

Slide 11: The scatterplot for these 2 variables are as shown. Temp has a slightly better correlation, maybe we can give it a shot at predicting count later on.

Slide 12: In the case of the categorical variables, some had only two categories, while some had more than two. For boolean variables, the Point Biserial Correlation Coefficient, or PBCC for

short, is an appropriate measure of correlation, whereas for variables with more than two categories, the Phi Coefficient is a more appropriate measure. From our preliminary data exploration earlier, we identified ‘holiday’ and ‘workingday’ to be the boolean variables, thus we used PBCC to measure their correlation with ‘count’. On the other hand, 'season' and 'weather' have more than two categories, hence we used Phi Coefficient to measure their correlation with 'count'.

Slide 13:
Here is the code we used to implement this and these are the corresponding correlations and p-values. p-value ranges between 0 to 1. A p-value of 1 indicates that there is almost certainly no significant relationship between the 2 values. It suggests that any observed correlation is likely due to chance and is not statistically significant. A p-value of 0 indicates that the observed correlation is unlikely to be due to chance and is statistically significant. So, in our case, we want to pick out variables that have high correlations with low p-values.

From the results, we can see that ‘weather’ has a p-value of almost 1, which means there is almost certainly no significant relationship between weather and count, and that the observed correlation is highly likely due to chance. We observe ‘season’, ‘month’ and ‘hour’ to have high Phi-Coefficients with low p-values. Thus, we will use these 3 categorical variables in our prediction.

Now that we have identified temp, season, month and hour to be correlated variables for our prediction, we tried using different plots to visualize them again.

Slide 14: To visualize the correlated variables, we used 2 types of plots. Hexbin plot and boxplot. Hexbin plot was chosen to display the relation of temperature and count as you can see how the count density changes as temperature increases or decreases. The high density of points in a particular region which is denoted by the darker colour suggest there is a strong relation between the 2 variables in that region.

Slide 15: From the boxplot of categorical values vs count, the only thing we can take from it is that count is significantly lower in spring compared to other seasons. Since each day in every season has its own ‘hours’, we felt visualizing it in a time series might help.

Slide 16: From the timeseries of user count against hours of the day for each season, you can see that spring indeed has the lowest count as its time series line is lower than all other seasons consistently. We can also see that count spikes at 0800 and 1700 hours across all seasons, perhaps this has something to do with the start and end of work or school.

Slide 17: As identified previously, our target variable count has a highly skewed distribution. Before we move to model fitting, we tried to normalize that distribution first. We tested log, square-root and box-cox transformation. After trying all, we came to the conclusion that using log gave the most normalized distribution.

Slide 18: Also, from the time series earlier, since we saw that spring has a significantly lower count across all timings and that the count of the other seasons were relatively similar, we decided to One Hot encode our season variable, using this code. This turns our season variable into 1 or 0, indicating whether it is spring or not spring respectively.

Slide 19: Now we are ready to move on to model fitting.
For our problem, we decided to try multiple models and then check the goodness of the fit to see which one fits our dataset the best. We used RandomForestRegressor, AdaBoostRegressor, BaggingRegressor, SupportVectorRegressor and KNeighboursRegressor. Here are the brief descriptions for each model.

Slide 20: This table shows the goodness of fit of the various models. As can be seen, KNeighbours has the lowest RMSLE. RMSLE (Root Mean Squared Log Error) - measures the ratio between the actual and predicted values. It is calculated as the square root of the mean of the squared logarithmic differences between the predicted and actual values. This model also has the lowest Mean absolute error, lowest root mean squared error, and highest variance. Hence, KNeighbours is the best suited model for this problem.

Slide 21: After finding that KNeighbours is the best model based on the error metrics, we want to perform hyperparameter optimization, which is the process of finding the right combination of hyperparameter values to achieve maximum performance on the data in a reasonable amount of time. This will improve the prediction accuracy of our machine learning algorithm.

To do so, we used this code here which uses grid search to find the optimal value of the hyperparameter n_neighbors for the KNN model. The n_neighbors parameter determines the number of neighbors to consider when making a prediction. The code uses a list of values to search over (in this case, [5, 10, 15, 20, 25, 30, 35, 40, 45])

The output {'n_jobs': -1, 'n_neighbors': 15} indicates that the optimal hyperparameters for the KNN model, as determined by the grid search, are n_jobs=-1 and n_neighbors=15.

n_neighbors=15 means that the KNN algorithm will consider the 15 closest neighbors to the test point when making a prediction. This hyperparameter was chosen as the best based on the grid search results and cross-validation performance.

Slide 22: Since our dataset has 10000 points and a test size of 2000 (0.2), a scatter plot may not be the best way to visualize the relationship between true and predicted values. Too many points make the graph crowded and hence, hard to analyze the trend.

As such, we came up with a histogram of the difference between true and predicted values.his helps us to visualize the distribution of errors and identify any patterns or biases in the predictions. As we can see, the model is able to keep a high frequency of predictions with low error, as evidenced by the high concentration of errors near zero in the middle of the histogram.

Slide 23: Overall, our findings have revealed that the usage of Kneighbours is an applicable model that is fairly accurate after some parameter tuning for companies to be able to improve user experience as our findings would allow them to predict when more bikes are needed to be supplied to meet user demand depending on different circumstances which are the correlated variables we identified. These circumstances, being the variables temperature, season, month and hour are applicable to every country apart from maybe non-temperate countries where there are no seasons. As such, we feel that our findings would have a wide scope of places it could be applied to.

Slide 24: Thank You!