Cover sheet for submission of work for assessment

SWINBURNE UNIVERSITY OF TECHNOLOGY

UNIT DETAILS

Unit name	Introduction to Data Science			Class day/time	Wed 10.30
Unit code	COS10022	Assignment no.	3	Due date	20 June 2020
Name of lecturer/teacher	Pei-Wei Tsai				
Tutor/marker's name	Pei-Wei Tsai				

Office use only

Faculty or school date stamp

STUDENT(S)

	Family Name(s)	Given Name(s)	Student ID Number(s)
(1)	Ridwan	Janice Graciela	101994340
(2)			
(3)			
(4)			
(5)			
(6)			

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Janice Graciela Ridwan	(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/**

INTRODUCTION

The dataset for this assignment from the UCI Machine Learning Database includes occupancy and capacity of various carparks in Birmingham, UK. Each observation includes date and time, from October 4 to December 19, 2016 and has times ranging from 8:00AM to 4:30PM. The observations were taken every 30 minutes.

The aim of this project is to visualise the carpark occupation rates to see which ones require expansion first. These would be the carparks that often reach full capacity. This is important because people spend a lot of time looking for parking spaces, and carparks that reach their full capacity often will potentially lose out on patrons that may go elsewhere as a result of the lack of parking spaces available.

For example, in the USA, 'motorists spend an average of 17 hours a year searching for spots on streets, in lots, or in garages' (McCoy, 2017). Not only is this frustrating for motorists, 'All this time looking for parking is not only counter-productive and impactful to the economy, but it is also creating more traffic' (Parkhound, 2014). Hence it is important for companies to invest into their carparks that are lacking in capacity. This visualisation and prediction will be done in Jupyter Notebook, using Python.

DATA VISUALISATION

DATA EXPLORATION AND ANALYSIS

To view some information of the dataset, I used the .head method. The attributes of the dataset are given in Table 1.

Attribute	Description
SystemCodeNumber	ID of each carpark
Capacity	The number of parking spots available in the carpark
Occupancy	The number of parkings spots occupied at the time recorded
LastUpdated	The date and time of the observation

Table 1 Attributes of dataset

Using len() function, we can see that there are 35,717 observations. We have to make sure that our data is in the expected format. For this dataset, we want the SystemCodeNumber to be a string (object in Pandas). Capacity and Occupancy should be integers, and LastUpdated should be a datetime. To check the data type, I used .dtypes. Figure 1 displays the results. As shown, the LastUpdated field is formatted as a string instead of a datetime. This will be fixed in the next step.

```
df_raw.dtypes
```

```
SystemCodeNumber    object
Capacity             int64
Occupancy            int64
LastUpdated         object
dtype: object
```

Figure 1 Attribute data types

DATA CLEANING

We already know that we have to change LastUpdated to be a datetime. This is achieved using .astype('datetime64'). Next, I decided to add some features to aid with the analysis (Table 2).

Attribute	Description
OccupationRate	The percentage ratio of Occupancy to Capacity.
Date	The date component of Date_Time_HalfHour.
Date_Time_HalfHour	This field rounds each time in LastUpdated to the nearest half hour.
Time	Only the time component of the Date_Time_HalfHour.

Table 2 Additional features

```
df_clean.groupby('time').size()
```

```
time
07:30:00      30
08:00:00    2096
08:30:00    1971
09:00:00    1953
09:30:00    1983
10:00:00    1987
10:30:00    1985
11:00:00    1961
11:30:00    1988
12:00:00    1976
12:30:00    1982
13:00:00    1988
13:30:00    1958
14:00:00    1984
14:30:00    1989
15:00:00    1986
15:30:00    1985
16:00:00    1956
16:30:00    1959
dtype: int64
```

Figure 2 Grouping by time

Grouping by time, as shown in Figure 2, we can see that 7:30 has only 30 rows which is very little compared to the rest. Hence, I decided to drop the rows observed at 7:30. Afterwards, we also want to remove any duplicate records if there are any. This is done simply using .drop_duplicates(). Checking the length of the resulting dataframe, we see that 207 records were duplicates and dropped.Next, we have to make sure OccupationRate is only between 0-100%, as anything outside of that must be due to faulty equipment or any errors. Checking the current data, we see the minimum is -1.67% and maximum is 104.13%.

```
print('Minimum Percent Occupied: {:}'.format(df_clean.OccupationRate.min()))
print('Maximum Percent Occupied: {:}' .format(df_clean.OccupationRate.max()))
```

```
Minimum Percent Occupied: -1.6666666666666667
Maximum Percent Occupied: 104.1343669250646
```

Figure 3 Percent Occupied

This is outside the range we want hence we will limit the Occupancy to the range of 0 to Capacity. Shown below, the occupation rate is cleaned and rounded to the nearest ten. It is rounded because OccupationRate will be used in a regression model, hence discrete values are required.

```
# Limit Occupancy to the range of zero to Capacity
df_clean.Occupancy = df_clean.apply(lambda x: max(0, min(x['Capacity'], x['Occupancy'])), axis=1)
df_clean['OccupationRate'] = df_clean.Occupancy / df_clean.Capacity * 100
df_clean['OccupationRate'] = round(df_clean['OccupationRate'].astype(int), -1)

# Re-check range
print('Minimum Percent Occupied: {:}'.format(df_clean.OccupationRate.min()))
print('Maximum Percent Occupied: {:}'.format(df_clean.OccupationRate.max()))
```

```
Minimum Percent Occupied: 0
Maximum Percent Occupied: 100
```

Figure 4 Percent Occupied after cleaning

DATA VISUALISATION

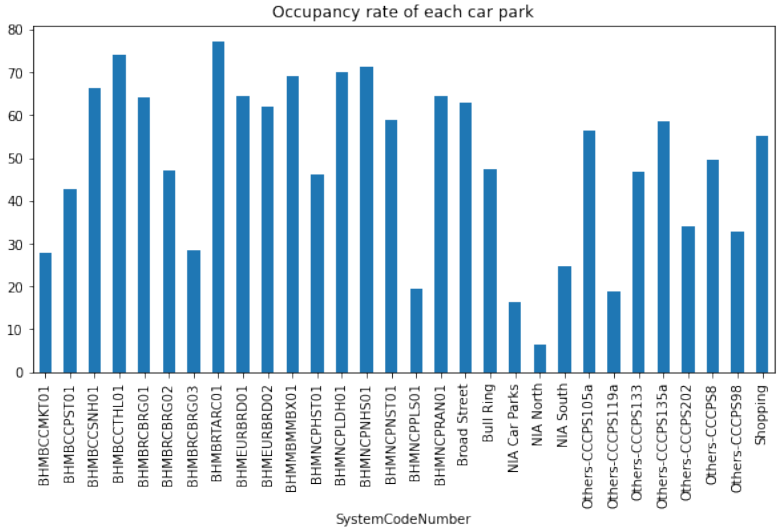

Figure 5 Carparks occupancy rate

Using the means of occupation rate of each carpark, a bar chart is produced. The bar chart shows the occupation rate of each carpark, but this is only an overview. To compare the carparks further, I visualised each with their daily occupancies as a line chart. Afterwards, I found two that had similar trends. These are BHMBCCMKT01 and BHMBCCTHL01. Below is the daily and weekly mean occupation rate of each chosen carpark.

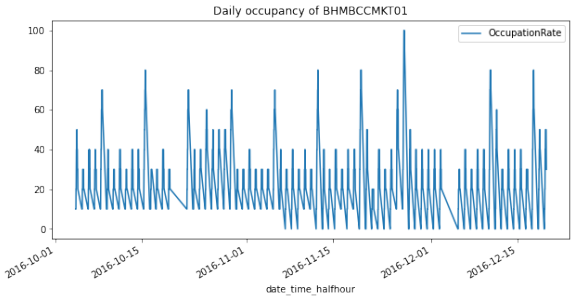

Figure 6

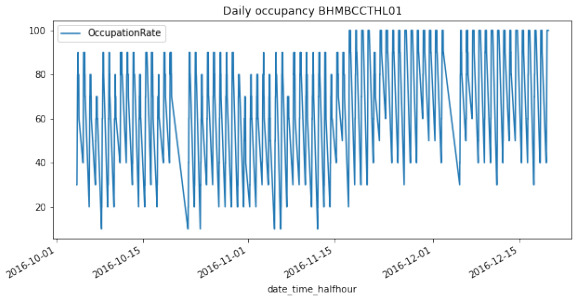

Figure 7

Figure 6 and 7 shows the daily occupancy of both carparks. The trends between the daily occupancy of two carparks are similar. This makes sense, as carparks will be fuller during weekdays if the carpark is located near office buildings, or if it is located near restaurants or cafes, which people will go to during lunch hours on weekdays. It is apparent that BHMBCCTHL01 peaks to 100% occupancy quite often nearing December and within December, but this may be because it is near the Christmas period and that the carpark is located near a shopping centre. From Figure 6 and 7, we can see that each week has trends that are about the same. To view the weekly occupation rate, we move to Figures 8 and 9 below.

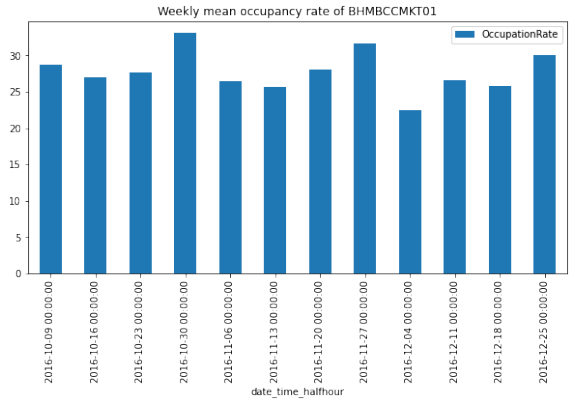

Figure 8

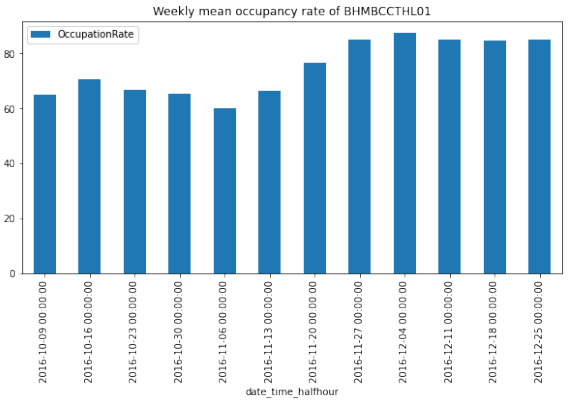

Figure 9

The weekly occupancy of each carpark are all around the same height compared to the other bars, although BHMBCCMKT01 seems to have occupation rate around 30-40% every week as seen in Figure 8 and BHMBCCTHL01 has it much higher, ranging from 60% to above 80%, in Figure 9. As we don't know where each carpark is located, it is difficult to tell why one carpark has a higher average occupancy rate every week as compared to the other. My speculation would be that BHMBCCTHL01 is in a higher traffic area, perhaps near large office buildings or highly frequented shopping centres, whereas BHMBCCMKT01 might be at smaller shopping centres that is often visited by people who live nearby.

CORRELATION ANALYSIS

Satisfied with the selected pair from the visualisation, I proceeded to perform a correlation analysis on the two. This was done by splitting them by groups, and then joining the two groups into a dataframe. This dataframe is then pivoted with the index of "date_time_halfhour", the columns "SystemCodeNumber" and values "OccupationRate" and "ndate_time_halfhour" to perform the correlation and to be used for the prediction model. "ndate_time_halfhour" is the numeric form of the attribute 'date_time_halfhour'. The pivot table obtained is given in Figure 10.

	OccupationRate		ndate_time_halfhour	
SystemCodeNumber	BHMBCCMKT01	BHMBCCTHL01	BHMBCCMKT01	BHMBCCTHL01
date_time_halfhour				
2016-10-04 08:00:00	10	30	1475568000000000000	1475568000000000000
2016-10-04 08:30:00	10	30	1475569800000000000	1475569800000000000
2016-10-04 09:00:00	10	40	1475571600000000000	1475571600000000000
2016-10-04 09:30:00	20	50	1475573400000000000	1475573400000000000
2016-10-04 10:00:00	20	60	1475575200000000000	1475575200000000000

Figure 10 Pivot Table

		OccupationRate	
	SystemCodeNumber	BHMBCCMKT01	BHMBCCTHL01
	SystemCodeNumber		
OccupationRate	BHMBCCMKT01	1.000000	0.631043
	BHMBCCTHL01	0.631043	1.000000

Figure 11 Correlation

With this pivot table, I can now use .corr() to obtain the correlation between the occupancy rate of the two carparks, which is 0.631043. "The greater the absolute value of the correlation coefficient, the stronger the relationship." (Frost, 2020) The correlation coefficient obtained is greater than 0.6, which is often classified as either a moderately strong or strong relationship. With this in mind, it can be concluded that the relationship between the two carparks is strong, which makes them appropriate to use as a pair.

MODEL BUILDING AND PREDICTION

DATA PARTITIONING

The analytic goal of this report is use carpark BHMBCCTHL01 to predict carpark BHMBCCMKT01 and vice versa. This is a regression problem hence I will evaluate some regression models and build the best option. To build the model, I must perform data partitioning first. The target variable, OccupationRate of BHMBCCMKT01 is in y, while X has the predictors. The ndate_time_halfhour of the target carpark is also dropped as we want to use the other carpark to predict carpark BHMBCCMKT01.

```
# Creating test and train models
#using X to predict label Y
X = piv_join.drop([('OccupationRate','BHMBCCMKT01'), ('ndate_time_halfhour', 'BHMBCCMKT01')], axis=1)
y = piv_join[('OccupationRate', 'BHMBCCMKT01')]
# y = y.astype(int)
print(list(piv_join.columns))
y
```

Figure 12 predictors and target variables

After splitting the attributes into target and predictors, I used train_test_split to partition the data. For my first iteration, I used a training and test size of 50% and use a random seed in order to be able to reproduce the training and test sets. Using the train test split, I will evaluate some regression models using k-fold cross validation to see which one will perform best. The models chosen are Linear Regression (LR), KNeighbors Regressor (KNN), Decision Tree Regressor (DTR), Random Forest Regressor (RFR).

```
# train test split 1
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.5,test_size=0.5, random_state = 10)
```

Figure 11 test train split

MODEL SELECTION

LINEAR REGRESSION

Linear Regression (LR) is a parametric method that attempts to fit a linear equation to observed data to model the relationship between two variables. In this case it fits the occupancy rate of BHMBCCTHL01 to observe the occupancy rate of BHMBCCMKT01 and vice versa. Linear regression has advantages such as smooth calculation, no adjustment parameters, and convenience. However, it has drawbacks such as a less accurate prediction compared to more complex models and it isn't possible to perform if the data has nonlinear features.

KNEIGHBORS REGRESSOR

In KNN regression, the target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. The output in KNN regression is the property value for the object which is the average of the values of *k* nearest neighbors. 'K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure.' (Muhajir, 2019)

Like linear regression, KNN is also one of the simplest models and differs from LR by being a non-parametric method, meaning that it doesn't assume a specific form for the function, f(X). This creates the advantage for KNN to be a more flexible approach compared to parametric methods, but it tends to be more complex to interpret as compared to parametric methods like LR.

DECISION TREE REGRESSOR

Decision tree builds models in the form of a tree structure by breaking down a dataset into smaller and smaller subsets while an associated decision tree is incrementally developed. This results in a tree with decision and leaf nodes. A decision tree is used for both regression and classification models, but in our case, it will be used as a regressor.

A decision tree is meant to capture the training data in the smallest possible tree. The reason for doing this is to use the simplest possible explanation, which produce decisions faster than larger, more complex trees. They are also easier to view, understand and interpret. Disadvantages of decision trees are that they are sensitive to the data they are trained on, meaning if the training data is changed the resulting decision tree can differ, hence predictions can change significantly. Decision trees are also expensive to train and carry a big risk of overfitting.

RANDOM FOREST REGRESSOR

With the drawbacks of decision trees, we turn to Random Forest. The trees in a random forest are run parallel to each other, meaning there isn't any interaction between these trees while building them. The algorithm works by constructing many decision trees at training and outputting the class that, in the case of regression, is the mean prediction of the individual trees. 'The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.' (Krishni, 2018)

Random Forest is one of the most accurate learning algorithms and runs well on large datasets. It can estimate missing data and maintains accuracy when a big part of the data is missing. Like decision trees, random forests may overfit datasets with regression tasks that have a lot of noise.

EVALUATING MODELS

'Cross validation (CV) is one of the techniques used to test the effectiveness of a machine learning models' (Sanjay.M, 2019). To cross validate, several metrics were used to verify how well each model performed. The metrics along with the reason they are chosen are given below.

- Mean Absolute Error: MAE is the average value of the absolute error, which can preferably reflect the actual situation of the predicted error. For MAE, the smaller it is, the better. 'MAE is a more natural measure of average error, and (unlike RMSE) is unambiguous.' (Willmott, et al. 2005)
- Root Mean Squared Error: RMSE is the arithmetic square root of MSE; which is the expected value of the square of the difference between the estimated and real parameter value. It is used to measure the deviation between observations and real values. The smaller the value of RMSE is, the more accurate the prediction model is. 'The 'squared' nature of this metric helps to deliver more robust results which prevents cancelling the positive and negative error values.' (Srivastava, 2019)

- r^2 score: r-squared score measures the goodness-of-fit of regression models and ranges from 0 to 1. Higher values represent smaller differences between the observed data and the fitted values, so r^2 values closer to 1 is preferred. Although r^2 scores are intuitive, they have their shortcomings. As 'the value of a model is generally in its overall accuracy and precision, not how successfully it explains the variation in a particular data set' (Alexander, et al. 2015), RMSE and MAE are better metrics for evaluation. Hence the r^2 values obtained will be used as supplementary scores.

To perform n-fold cross validation with these metrics, I looped through each regression model with cross_val_score on the training set, to produce their MAE, RMSE and r^2 scores. The bar chart below presents the results. The code used to do this is in Figure 12.

```
nfolds=10
i=0
results = []
for name, model in models:
    kf = KFold(n_splits=nfolds)
    score = cross_val_score(model, X_train, y_train, cv=nfolds, scoring= 'neg_root_mean_squared_error')
    score2 = cross_val_score(model, X_train, y_train, cv=nfolds, scoring= 'r2')
    score3 = cross_val_score(model, X_train, y_train, cv=nfolds, scoring= 'neg_mean_absolute_error')
    results.append((name, score))
    if score.mean() > i:
        imodel=model
        i=score.mean()
    print(name, score.mean(), score2.mean(), score3.mean())
print("Done")
```

Figure 12 n-fold cross validation

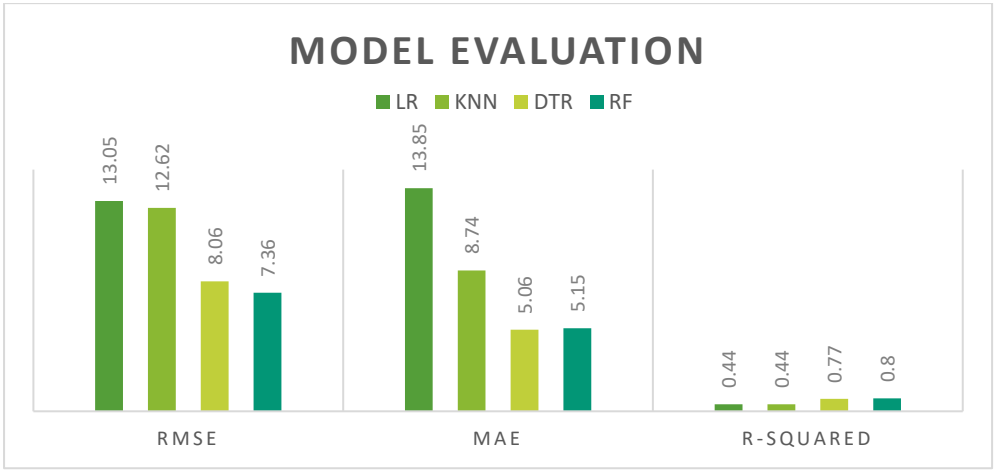

Figure 13 Model Cross Evaluation scores

Keeping in mind that lower RMSE and MAE values are desirable, and lower r^2 scores, Decision Tree Regressor and Random Forest Regressor are the best options as presented in Figure 13. Seeing as Random Forest evaluated better in both RMSE and r^2, and only has a 0.09 difference in MAE, we will go with a Random Forest model in this project.

BUILDING THE RANDOM FOREST MODEL

```
#First RF model

clf=RandomForestRegressor(max_depth = 10, min_samples_split=2, n_estimators = 50, random_state = 10)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)


print("Mean Absolute Error (MAE) : {}".format(mean_absolute_error(y_test, y_pred)))
print("Root Mean Squared Error (RMSE) : {}".format(rmse(y_test, y_pred)))
print("r2 Score : {}".format(r2 score(y test, y pred,multioutput='variance weighted')))
```

Figure 14 Regression Model

With the Random Forest Regressor, I trained the model with the training set and then to validate the accuracy of the model, I tested it on the test set. The accuracy is evaluated using the same metrics used to cross validate the different models; MAE, RMSE and r^2 scores. The values obtained are given in Table 3.

Metric	Score
MAE	5.154
RMSE	7.321
r^2	0.822

Table 3 Evaluation Metric Scores

An r^2 score of 0.822 is good as it tells us that 82% of the variance of the target variable can be explained by the predictors. The RMSE and MAE scores at 7.3 and 5.2 respectively are decent seeing as the occupation rate ranges from 0 to 100.

FINE TUNING

The results are already quite good, but there might be a way to improve the model's performance by fine tuning. First, I tried to change the parameters of the regressor. The parameters changed were max_depth and n_estimators. I found that max_depth of 20 is best, and n_estimators of 100 is ideal. These are good as they improve the model but will not cause it to be significantly slower. Figure 15 displays the code used.

```
clf=RandomForestRegressor(max_depth = 20, min_samples_split=2, n_estimators = 100, random_state = 20)
```

Figure 15 Fine Tuning parameters

Then, I decided to change the train test split into a train size of 80%, and test size of 20%. Using this partition and the model with the changed parameters, the evaluation matrices show that there is an improvement in the model's performance, with MAE improving by 19.4%, RMSE by 18.6% and r^2 by 6.9%. The percentage changes in MAE and RMSE show us that the fine tuning significantly improved the model.

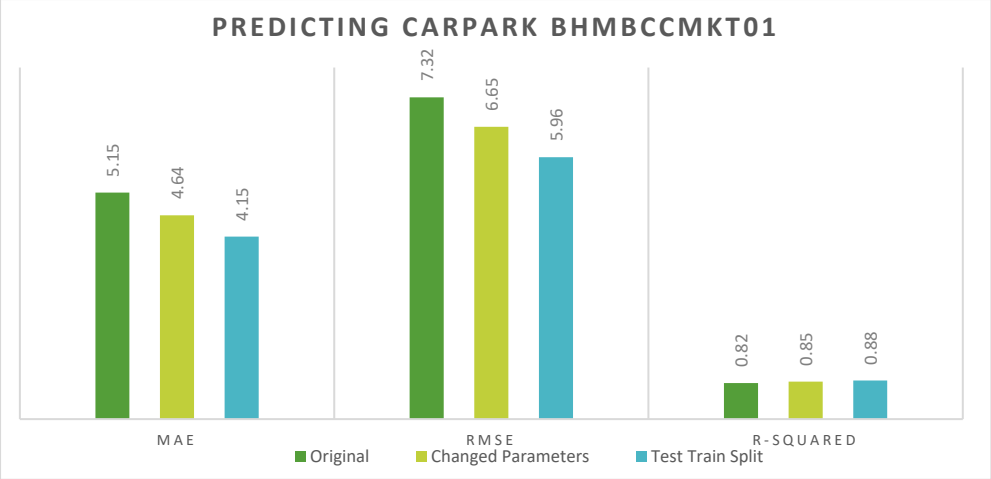

Figure 16 Improvement in model for first prediction

FINAL PREDICTION MODEL

To predict the second carpark, I made carpark BHMBCCTHL01 as the target variable and repeated the same process as previous. To see if the fine tuning worked on the model, I tested the original model and the final model on this carpark as well. The evaluation metrics gives similar improvements, MAE improved by 21.55, RMSE by 15.7% and r^2 by only 3.4%. As the r^2 value is only supplementary, judging by the MAE and RMSE we can verify that changing parameters and using a larger training set improved the model.

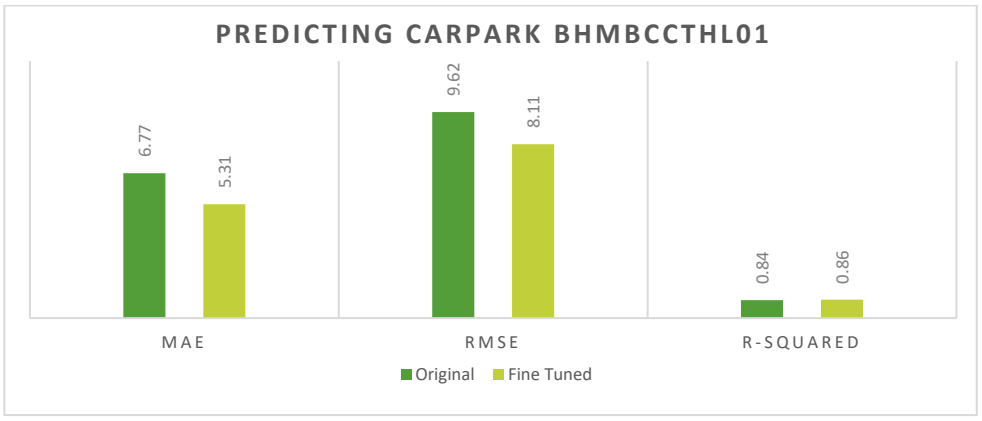

Figure 17 Improvement in model for first prediction

The evaluation matrices obtained for predicting the occupation rates of carpark BHMBCCMKT01 and BHMBCCTHL01 with a random forest regressor are as follows:

Carpark	MAE	RMSE	r-squared
BHMBCCMKT01	4.15	5.96	0.88
BHMBCCTHL01	5.31	8.11	0.86

Table 4 Both Carparks Prediction Evaluation

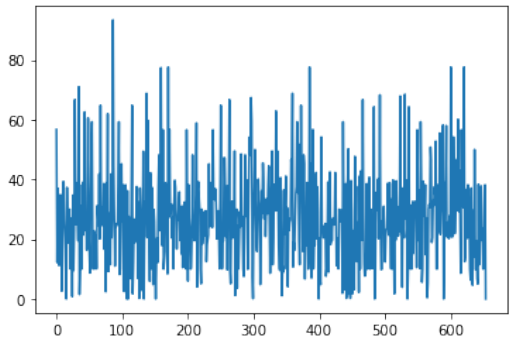

Figure 10 BHMBCCTHL01 prediction results

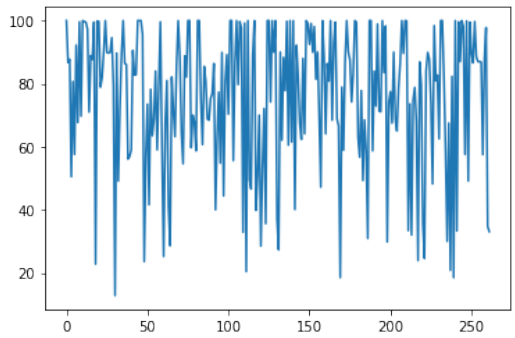

Figure 11 BHBCCMKT01 prediction results

Figures 10 and 11 show the prediction results of the model. These figures are reminiscent of the daily occupancy rates of each carpark given in the visualisation, Figures 6 and 7. The mean prediction results for each carpark BHMBCCMKT01 and BHMBCCTHL01 are 27.4 and 75.8% respectively. These results are close to the average values shown in the weekly mean occupancy rates in Figure 8 and 9. This confirms that the prediction model built for the analytic goal of predicting the occupation rate of carpark BHMBCCMKT01 with carpark BHMBCCTHL01 and vice versa is appropriate.

EXPANSION

Although there is sufficient information to build a prediction model on this dataset, there are attributes that would aid in building a more accurate model in the future. The information we know about each carpark is limited to its capacity, which does not give us a clue as to why the occupation rates may differ so greatly. I have made assumptions in the visualisation section of this report, however, they are just assumptions. Hence, an attribute that would be of great use is the location of the carpark, whether it is in a more metropolitan area or suburban parts of the city. An example would be the data from a similar study, where data 'is captured every 5 minutes including datetime, geolocation(latitude, longitude), place name, lot capacity, occupied number of spots, parking price, lot type (on/off street parking).' (Chen, 2014)

Knowing the location of the carparks would help explain the occupation rates while visualising them. For example, metropolitan areas are typically higher traffic hence the carparks there would have similar, higher occupation rates than others. These carparks that are within a certain radius could be aggregated and compared against carparks in other locations. So, we take carparks in two locations instead of taking two carparks to compare against each other. The benefit of this approach is a reduction in the prediction error; however, it will most likely decrease after the aggregation of carparks reaches a certain number.

Other features that may be a useful addition to the dataset would be the weather and population density, which highly affect the number of cars as people may be more likely to drive to work on rainy days, and the higher the population density, the higher the chances of there being more cars. However, all these attributes are limited to the amount of time the data was observed. As there was only 10 weeks' worth of data, the predictions will not be as accurate and the model will not be able to be trained as well as if the data collection was over the course of a year or more.

REFERENCES

Frost, J 2020, *Interpreting Correlation Coefficients*, Statistics By Jim, viewed 15 June 2020, <https://statisticsbyjim.com/basics/correlations/>

Chen, X 2014, *Parking Occupancy Prediction and Pattern Analysis*, Semantics Scholar, viewed 16 June 2020, <https://www.semanticscholar.org/paper/Parking-Occupancy-Prediction-and-Pattern-Analysis-markcx/65efc2f9c2c9e8675426427c765086974d6c920d>

Parkhound, 2014, *Australian drivers spend over 3,000 hours looking for parking in their lifetime*, Parkhound, viewed 16 June 2020, <https://www.parkhound.com.au/blog/australian-drivers-spend-over-3000-hours-looking-for-parking-in-their-lifetime/>

McCoy, 2017, *Drivers spend an average of 17 hours a year searching for parking spots*, USA Today, viewed 16 June 2020, <https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/>

Sanjay.M, 2018, *Why and how to Cross Validate a Model?*, Towards Data Science, viewed 16 June 2020, <https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f>

Srivastava, T 2019, *Important Model Evaluation Metrics for Machine Learning Everyone should know*, Analytics Vidhya, viewed 17 June 2020, <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>

Willmott, CJ, Matsuura, K 2005, *Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance*, Research Gate, viewed 17 June 2020, <https://www.researchgate.net/publication/235710066_Advantages_of_the_Mean_Absolute_Error_MAE_over_the_Root_Mean_Square_Error_RMSE_in_Assessing_Average_Model_Performance>

Alexander, D, Tropsha, A, Winkler, D 2015, *Beware of R^2: simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models*, National Library of Medicine National Institutes of Health, viewed 18 June 2020, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4530125/>

Krishni, 2018, *A Beginners Guide to Random Forest Regression*, Medium, viewed 16 June 2020, <https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb>

Muhajir, 2019, *K-Neighbors Regression Analysis in Python*, Medium, viewed 16 June 2020, <https://medium.com/analytics-vidhya/k-neighbors-regression-analysis-in-python-61532d56d8e4>