COMP4432 Machine Learning - Assignment 4

Learning Objective: This assignment will allow the construction of multiple regression models to predict bike share usage. After exploring, analyzing, and preparing the data, multiple models will be constructed to estimate the target variable (the *cnt* variable in the data set). The data includes two years of hourly bike share usage, and includes details on:

- weather conditions:
  - temperature
  - humidity
  - wind
  - weather situation
- time of usage details:
  - time of day
  - day of week and month
  - month
  - year,
  - season
  - flags to identify weekdays and holidays

Part 1: Data Exploration and Analysis

a) The data set for Assignment 4 is located on GitHub at:
https://raw.githubusercontent.com/arjayit/cs4432_data/master/bike_share_hour.csv
Details and descriptions of the data set can be reviewed in the README file at:
https://github.com/arjayit/cs4432_data/blob/master/bike_share_Readme.txt
Please review the *Background* and *Data Set* sections of the README file.
Import *bike_share_hour.csv* as a Pandas DataFrame with the following code:
url = 'https://raw.githubusercontent.com/arjayit/cs4432_data/master/bike_share_hour.csv'
df = pd.read_csv(url)

b) The README file claims the data set consists of bike share usage reported hourly over two calendar years. Identify and print the number of unique date labels (*dteday*) in the DataFrame. In a Markdown cell, document how this value compares to the expected number of days in two years of data.
Hint: Examine the values of the two years present in the data. Is either special?

c) Execute the following code: *df.dteday.value_counts()*
The output shows the number of time each dteday value appears in the data set. For example, "2011-01-01" appears 24 times, and aligns with the hourly reporting.
If you are interested in some of the dates with a lot of missing data:
2012-10-29: Hurricane Sandy
2011-01-26 & 2011-01-27: North American Blizzard
2011-08-27 & 2011-08-28: Hurricane Irene
Now execute: *df.dteday.value_counts().value_counts()*
The output presents the number days for the given number of hours.
655 of the 731 days (≈ 90%) in the data have complete 24 hour reporting. The hourly data that is missing from the incomplete dates represents the hours that *cnt* is equal to zero.

d) Remove the *dteday* and *instant* input features. For this analysis, These features are purely descriptive.

e) Identify the categorical features, and save them to a list.
Hint 1: Calling either *df.dtypes* or *df.info()* will show that almost all input features are either *int* or *float* data types. Examine the descriptive statistics of the input data and review the data descriptions found in the README file.
Hint 2: **Eight** features will be categorical. Six of the eight are label encoded, and the remaining two describe components of time, so they are sequential, but will be encoded as categorical to give flexibility to model training.

f) Calculate the number of null values in the dataset.
Remove any rows with null values.

g) Construct code to map and display the values in *month* to the values in *season*. That is, identify which values in the *month* feature belong to each value in the *season* feature. In a Markdown cell, compare this month-to-season mapping to the data description for *season* in the Dataset Characteristics Section of the README file and document your findings.
Tip: Pandas groupby functions are helpful.
Hint: Yes, *Spring* is spelled *Springer*, but after reviewing the *season* description in the README file and the month-to-season mapping, you should find something else isn't correct.

h) Examine the descriptive analysis of the numeric input feature columns.
Review the data descriptions in the README file, and in a Markdown cell, comment on the ranges of the numerical features. Notice the target variable *cnt* has a minimum value of one, in agreement with the analysis in Part 1C.

i) Construct a bar plot of *cnt* versus *season.* Consider whether the sum of *cnt* or the mean of *cnt* per season is a better representation. In a Markdown cell, document any pattern observed. Does the graph agree with your intuition about bike usage and seasonality?

j) Construct a bar chart of *cnt* versus *workingday*. Consider whether the sum or the mean is a better representation. In Markdown cell, document how bike rides are distributed across the two options of working day.

k) Construct a bar chart for *cnt* versus *mnth.* Consider whether the sum or the mean is a better representation. In a Markdown cell, discuss the pattern. Does this agree with your intuition?

l) Construct a bar plot of *cnt* versus *weathersit.* Consider whether the sum or the mean is a better representation. In a Markdown cell, discuss any pattern.

m) Construct a point plot of *weathersit* on the x-axis, *cnt* on the y-axis, and *season* as the hue.
In a Markdown cell, document any relationship between ridership and *season* and *weathersit.*
The results of Part 1m and Part 1j might be helpful.

n) Construct a bar plot of *cnt* versus *hr* for *workingday = 1*. Consider whether the sum or the mean is a better representation. In a Markdown cell, discuss any pattern? Does this agree with your intuition?

o) Construct a bar plot of *cnt* versus *hr* for *workingday = 0.* Consider whether the sum or the mean is a better representation. In a Markdown cell, discuss the pattern, and how it compares to *workingday = 1*.

p) Construct a histogram of the *cnt* column.
In a Markdown cell, offer a brief description on the distribution. Recall, the target variable *cnt* represents the number of bike share usages per hour, and has a minimum value of one. Hours with no usage (*cnt* = 0) are not included in the data.

Part 2: Data Preparation

a) Calculate the correlation values for all columns (input features and target) in the dataset.
In a Markdown cell, document any highly correlated relationships, not just those with the target.

b) Drop the following columns from your dataset: *casual*, *registered*, and *atemp*.
atemp is highly correlated with *temp*, and while *casual* and *registered* are highly correlated to the target variable *cnt*, they sum together to give *cnt*.
Hint: As a sanity check, after removing the above columns, the entire DataFrame should have 12 columns (11 input features and the target variable).

c) Using *train_test_split* with a fixed random state for reproducibility, partition the dataset into training and testing data with a test size of 33%.

Part 3: Baseline Model Construction

a) Construct a baseline linear regression model using either *cross_val_score* or *cross_validate* methods from *sklearn.model_selection,* and the following:
- A Linear Regressor estimator (*sklearn.linear_model.LinearRegression*)
- Training data from Part 2C
- *r2_score* and *neg_root_mean_squared_error* evaluation metrics
- Five folds

b) Print the average R-Squared and average RMSE values over the 5 folds.
Hint: The scores obtained demonstrate the performance of a linear regressor without encoding the categorical features.

Part 4: Initial Model Training

a) The categorical features in the data from Part 2C will now be encoded. Using the *OneHotEcoder* functionality in *sklearn.preprocessing*, encode the identified categorical features from Part 1E in the Training and Testing data set.
Hint: Set *sparse_output = False* to get dense output.
Hint 2: Review the previously shared notebook: *one_hot_encoding.ipynb*
Hint 3: The encoded input data sets (X_train and X_test) should have 52 total columns.

b) Construct a baseline linear regression model using either *cross_val_score* or *cross_validate* methods from *sklearn.model_selection,* and the following:
- A Linear Regressor estimator (*sklearn.linear_model.LinearRegression*)
- The encoded Training data from Part 4A
- *r2_score* and *neg_root_mean_squared_error* evaluation metrics
- Five folds

c) Calculate the average R-Squared and RMSE values over the 5 folds in Part 4B.
In a Markdown cell, document how these scoring metric values compare to those in Part 3A. Did encoding the categorical features help model performance? The scores obtained here demonstrate the performance of a linear regressor after encoding the categorical features.

d) Similar to Parts 4B and 4C, using either *cross_val_score* or *cross_validate* methods with 5 folds, calculate the average R-Squared and average RMSE values for a default *Decision Tree Regressor* with *random_state=0* and with the encoded training data from Part 4A.

e) Similar to Parts 4B and 4C, using either *cross_val_score* or *cross_validate* methods with 5 folds, calculate the average R-Squared and average RMSE values for a default *Random Forest Regressor* with *random_state=0* and with the encoded training data from Part 4A.

f) Present a table that shows the average R-Squared and average RMSE values for the three algorithms used in Parts 4B-4E. These metrics reflect the performance of the algorithms with default hyperparameter settings.
Example of a table to show:

	Algorithm	Average R-Squared	Average RMSE
0	LinearRegression	0.680598	102.399634
1	DecisionTreeRegressor	0.804281	80.111438
2	RandomForestRegressor	0.899385	57.474476

Part 5: Model Tuning

a) From Part 4, and shown in the example table, the Random Forest Regressor should have the optimal values of R2 and RMSE.
To identify the optimal hyperparameters for the Random Forest Regressor, *RandomizedSearchCV* (*sklearn.model_selection*) will be used with the following settings:
- 20 iterations
- five folds
- *neg_root_mean_squared_error* scoring metric
- A fixed random state for reproducibility
- The encoded training data from Part 4B
- A single search over the following hyperparameter values:
  max_depth: 5, 10, 15, 20, 25, 30, 35, 40
  max_features: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
  n_estimators: 50, 100, 150, 200, 250

b) Print the optimal hyperparameters and RMSE score found from this search.

c) In a Markdown cell, document the comparison of RMSE metrics between the Random Forest with default hyperparameter settings in Part 4E and the Random Forest identified using Randomized Search in Part 5A. Did your randomized search return a more optimal Random Forest than using default settings? Furthermore, were the optimal hyperparameter settings identified with Randomized Search?

d) With the RandomForestRegressor from Part 5A, calculate the R-Squared Score and RMSE from predictions made with the encoded **Testing** data from Part 4A. How do these scores compare to the scores found in Part 4E (the default hyperparameter model)?