# block-2-coursework-cst2130

February 28, 2024

[1]:
```
#Rian Qadir M00975827

# Alvin Kihara M00709239

# Zia Sherchan M00906875

# Abinajan Rajan M00677206
```

[2]:
```
# Import necessary libraries
import pandas as pd                # For data manipulation using DataFrames
import matplotlib.pyplot as plt    # For data visualization
import seaborn as sns              # For statistical data visualization
import numpy as np                 # For numerical operations
# For splitting the data and cross-validation
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier #For the Random Forest␣
↪classifier
# For model evaluation
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from matplotlib.ticker import FuncFormatter  # For custom tick formatter in␣
↪matplotlib
from sklearn.preprocessing import OneHotEncoder # For one-hot encoding␣
↪categorical variables
from sklearn.compose import ColumnTransformer # For column transformations in a␣
↪pipeline
from sklearn.pipeline import Pipeline # For creating a pipeline of processing␣
↪steps
```

[3]:
```
# Read the dataset.
ldn_bikes = pd.read_csv('London_bike_data.csv')
```

[4]:
```
print ('First 10 rows of Data.')
pd.set_option('display.width', 1000)## resizing table so it doesn't split into␣
↪two (notice bike-data etc. is split into another table)
print(ldn_bikes.head(10))##printing table (max 10 results)
```

```
First 10 rows of Data.
      id        date  hour  season  is_weekend  is_holiday  temperature
temperature_feels  humidity  wind_speed  weather_code bike_rented
0   8650  2016-01-01     6       3           0           1          3.0
0.0       87.0        10.0                1    very low
1   9383  2016-01-31    19       3           1           0         14.0
14.0       77.0        35.0                3        low
2  12036  2016-05-22     8       0           1           0         14.5
14.5       65.0         6.5                1        low
3   2404  2015-04-14    11       0           0           0         18.0
18.0       54.0        21.5                1      medium
4   7406  2015-11-09    21       2           0           0         15.0
15.0       82.0        31.5                4      medium
5  16165  2016-11-12    22       2           1           0         11.0
11.0       88.0        13.0                4        low
6   6449  2015-09-30    16       2           0           0         17.0
17.0       42.0        31.0                1    very high
7   7331  2015-11-06    18       2           0           0         18.0
18.0       83.0        22.0                3    very high
8   3735  2015-06-09     1       1           0           0          9.0
7.0       76.0        14.0                1    very low
9   7223  2015-11-02     6       2           0           0          8.5
7.5       97.0         8.0                4        low
```

[5]:
```python
print ('First 10 rows of Data, in table form.')
ldn_bikes.head(10)## placing data into more organized table
```

```
First 10 rows of Data, in table form.
```

[5]:
```
      id        date  hour  season  is_weekend  is_holiday  temperature
temperature_feels  humidity  wind_speed  weather_code bike_rented
0   8650  2016-01-01     6       3           0           1          3.0
0.0       87.0        10.0                1    very low
1   9383  2016-01-31    19       3           1           0         14.0
14.0       77.0        35.0                3        low
2  12036  2016-05-22     8       0           1           0         14.5
14.5       65.0         6.5                1        low
3   2404  2015-04-14    11       0           0           0         18.0
18.0       54.0        21.5                1      medium
4   7406  2015-11-09    21       2           0           0         15.0
15.0       82.0        31.5                4      medium
5  16165  2016-11-12    22       2           1           0         11.0
11.0       88.0        13.0                4        low
6   6449  2015-09-30    16       2           0           0         17.0
17.0       42.0        31.0                1    very high
7   7331  2015-11-06    18       2           0           0         18.0
18.0       83.0        22.0                3    very high
```

```
8   3735  2015-06-09     1       1           0           0          9.0
7.0      76.0        14.0            1     very low
9   7223  2015-11-02     6       2           0           0          8.5
7.5      97.0         8.0            4         low
```

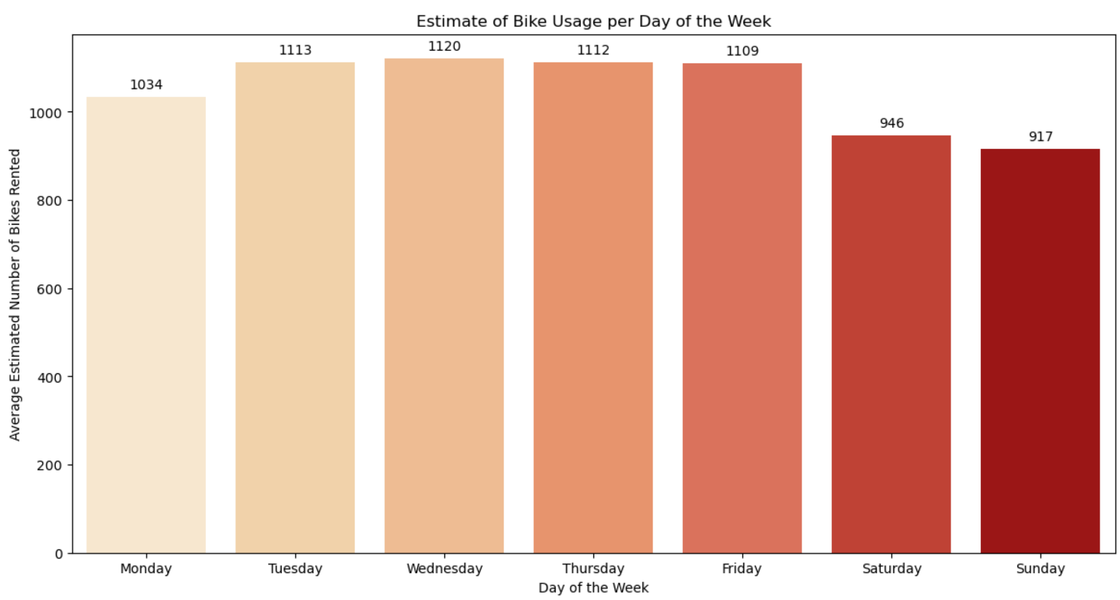
[6]:
```
#Assigning midpoints to 'bike_rented' categories
def calculate_midpoint(category):
    convert_to_numbers = {
        "very low": (1 + 169) / 2,
        "low": (170 + 600) / 2,
        "medium": (600 + 1100) / 2,
        "high": (1100 + 1900) / 2,
        "very high": (1900 + 3000) / 2
    }
    # Return the midpoint corresponding to the input 'category'
    return convert_to_numbers[category]
# Applying the calculate_midpoint function to create a new column␣
 ↪'estimate_bikes_rented'
ldn_bikes['estimate_bikes_rented'] = ldn_bikes['bike_rented'].
 ↪apply(calculate_midpoint)
```

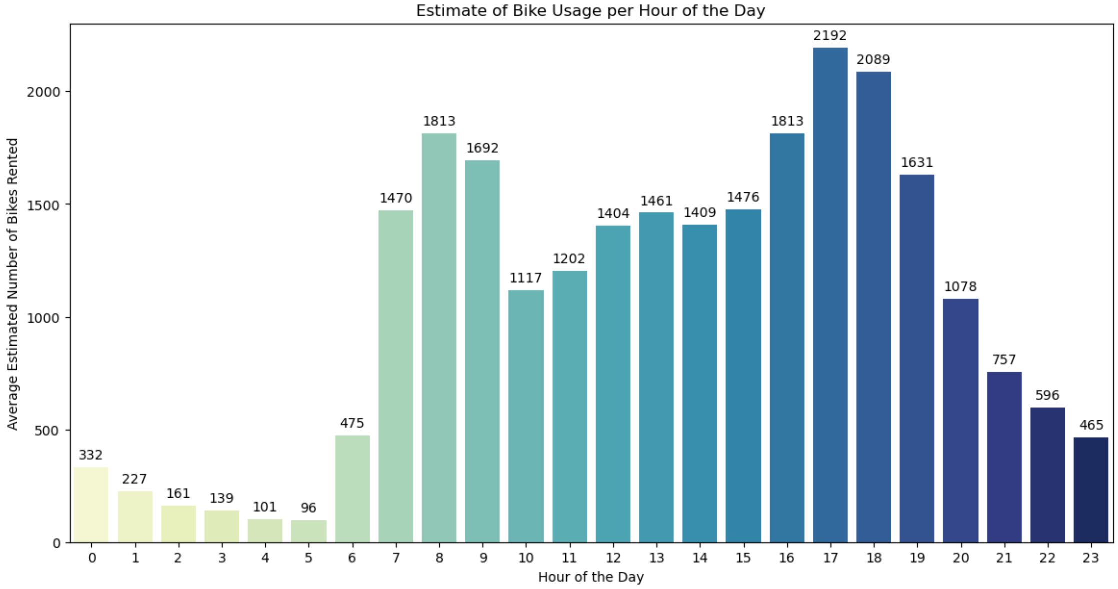
[7]:
```
# Convert 'date' to datetime and extract components
ldn_bikes['date'] = pd.to_datetime(ldn_bikes['date'])
ldn_bikes['day_of_week'] = ldn_bikes['date'].dt.dayofweek
ldn_bikes['hour'] = pd.to_numeric(ldn_bikes['hour'])

# # Calculate averages
daily_counts = ldn_bikes['date'].dt.date.nunique()
hourly_counts = daily_counts  # Using the number of unique days for hourly␣
 ↪calculation
daily_avg = ldn_bikes.groupby('day_of_week')['estimate_bikes_rented'].mean()
hourly_avg = ldn_bikes.groupby('hour')['estimate_bikes_rented'].mean()
```

[8]:
```
# Define the formatter for the y-axis to display integers
def formatter_integers(x, pos):
    return f'{int(x)}'

formatter = FuncFormatter(formatter_integers)
```

[9]:
```
# Plotting function
def def_plot(avg_data, title, x_label, y_label, palette, x_labels=None):
    plt.figure(figsize=(14, 7))
    # Create a bar plot using Seaborn
    plot = sns.barplot(x=avg_data.index, y=avg_data.values, palette=palette)
    # Format the y-axis labels using the specified formatter
    plot.yaxis.set_major_formatter(formatter)
    # Set the title, x-axis label, and y-axis label for the plot
```

```
    plot.set_title(title)
    plot.set_xlabel(x_label)
    plot.set_ylabel(y_label)
    # set custom x-axis tick labels
    if x_labels:
        plot.set_xticklabels(x_labels)
    for p in plot.patches:
        plot.annotate(format(p.get_height(), '.0f'),
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha='center', va='center',
                      xytext=(0, 9),
                      textcoords='offset points')
    plt.show()

# Generate the plots
days_of_week_labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
def_plot(daily_avg, 'Estimate of Bike Usage per Day of the Week', 'Day of the␣
 ↪Week', 'Average Estimated Number of Bikes Rented', "OrRd",␣
 ↪x_labels=days_of_week_labels)
def_plot(hourly_avg, 'Estimate of Bike Usage per Hour of the Day', 'Hour of the␣
 ↪Day', 'Average Estimated Number of Bikes Rented', "YlGnBu")
```






[10]:
```
# Check for missing values
print(ldn_bikes.isnull().sum())

# Convert categorical variables to numerical using one-hot encoding(binary)
ldn_bikes = pd.get_dummies(ldn_bikes, columns=['season', 'weather_code'])
```

```
id                      0
date                    0
hour                    0
season                  0
is_weekend              0
is_holiday              0
temperature             0
temperature_feels       0
humidity                0
wind_speed              0
weather_code            0
bike_rented             0
estimate_bikes_rented   0
day_of_week             0
dtype: int64
```

[11]:
```
print(ldn_bikes.columns) ## double checking columns
```

```
Index(['id', 'date', 'hour', 'is_weekend', 'is_holiday', 'temperature',
'temperature_feels', 'humidity', 'wind_speed', 'bike_rented',
'estimate_bikes_rented', 'day_of_week', 'season_0', 'season_1', 'season_2',
'season_3', 'weather_code_1', 'weather_code_2', 'weather_code_3',
```

```
'weather_code_4', 'weather_code_7', 'weather_code_10', 'weather_code_26'],
dtype='object')
```

[29]:
```
#contains the column names will be used as input features for the machine␣
↪learning model. These features will be used to predict the target variable␣
↪'bike_rented'.
features = ['hour', 'is_weekend', 'is_holiday', 'temperature',␣
↪'temperature_feels', 'humidity', 'wind_speed', 'season_0', 'season_1',␣
↪'season_2', 'season_3', 'weather_code_1', 'weather_code_2',␣
↪'weather_code_3', 'weather_code_4', 'weather_code_7',␣
↪'weather_code_10','weather_code_26']
#creates a new DataFrame X that includes only the columns specified in the␣
↪features list and is selected from the input features from the original␣
↪DataFrame Ldnbike
X = ldn_bikes[features]

#creates a Series y that represents the target variable. It extracts the␣
↪'bike_rented' column from the original DataFrame Ldnbike
y = ldn_bikes['bike_rented']
```

[33]:
```
# Split the data into training and testing sets using k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize the model
model = RandomForestClassifier()

# Train and evaluate the model and display the cross validation and mean␣
↪accuracy
accuracy_scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')

#displays the Cross-Validation Accuracy Scores
print("Cross-Validation Accuracy Scores:", accuracy_scores)

#displays Mean Accuracy
print("Mean Accuracy:", np.mean(accuracy_scores))
```

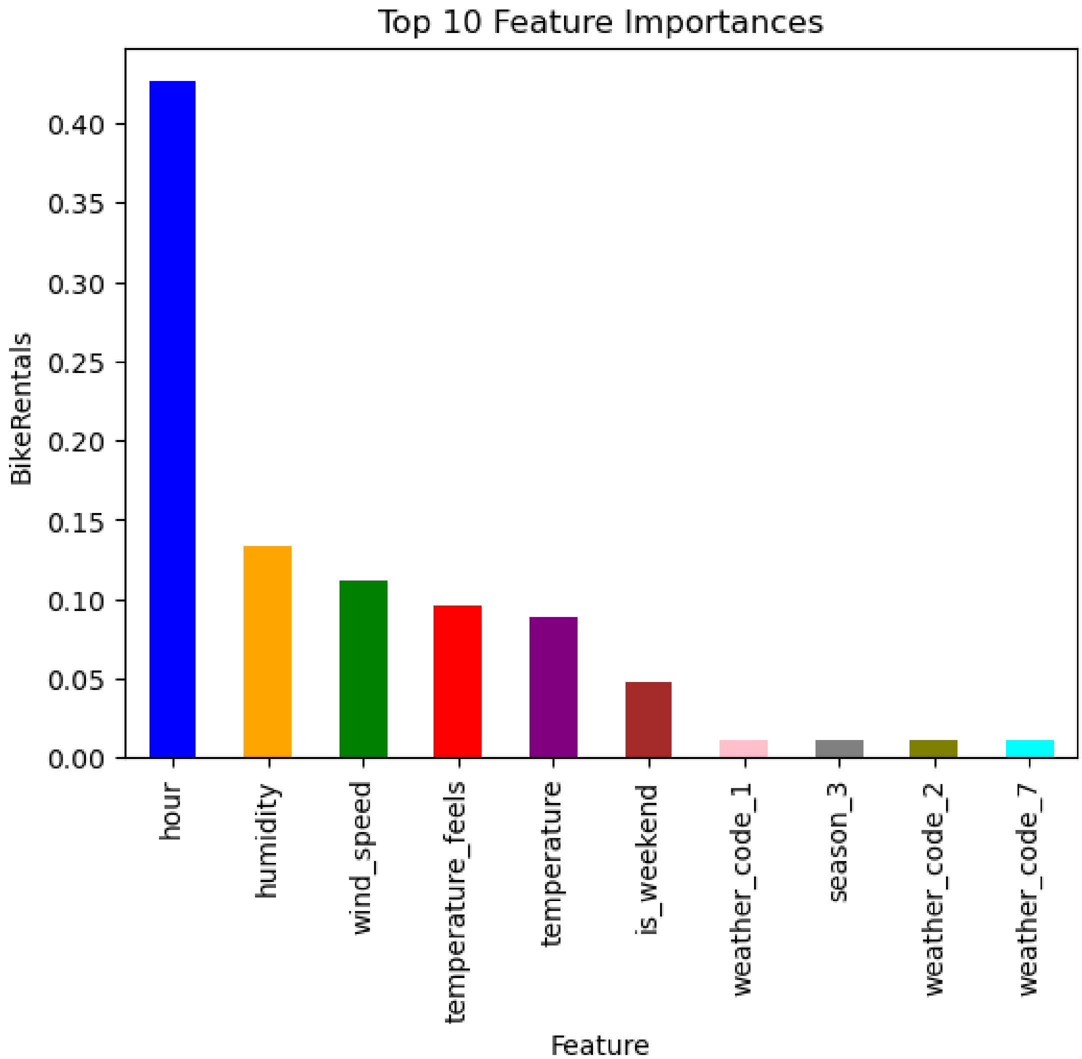
```
Cross-Validation Accuracy Scores: [0.78905054 0.7856049  0.77756508 0.77335375
 0.79287902]
Mean Accuracy: 0.7836906584992344
```

[45]:
```
# Train-test split for final evaluation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Fit the model on the training data
model.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model

# calculates the accuracy on the test set using the true labels (y_test) and␣
↪the predicted labels (y_pred) and then prints the result
print("Accuracy on Test Set:", accuracy_score(y_test, y_pred))

# calculates the classification report using the true labels (y_test) and the␣
↪predicted labels (y_pred) and then prints the result
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# the confusion matrix using the true labels (y_test) and the predicted labels␣
↪(y_pred) and then prints the result
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))


# Visualize feature importances

# creates a Pandas Series (feat_importances) from the feature importances␣
↪provided by the trained machine learning model
# feature_importances_ attribute in random forests, and it represents the␣
↪importance of each feature in making predictions
feat_importances = pd.Series(model.feature_importances_, index=features)

# selects the top 10 features with the largest importance values from the␣
↪feat_importances Series.
# The nlargest function is used to retrieve the specified number of largest␣
↪elements
top_feat_importances = feat_importances.nlargest(10)

# Define colors for the bars
colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown', 'pink', 'gray',␣
↪'olive', 'cyan']
# plot(kind='barh') creates a horizontal bar plot of features.
# The kind='barh' parameter specifies the type of plot (horizontal bar chart)
top_feat_importances.plot(kind='bar', color=colors)

# adds a title to the plot, indicating that it shows the top 10 features
plt.title("Top 10 Feature Importances")  # label of the graph
plt.xlabel("Feature")  # x axis label
plt.ylabel("BikeRentals")  # y axis label
plt.show()
```

```
Accuracy on Test Set: 0.7901990811638591
```

```
Classification Report:
              precision    recall  f1-score   support

        high       0.72      0.69      0.71       530
         low       0.79      0.77      0.78       527
      medium       0.65      0.70      0.67       471
   very high       0.87      0.85      0.86       538
    very low       0.91      0.92      0.92       546

    accuracy                           0.79      2612
   macro avg       0.79      0.79      0.79      2612
weighted avg       0.79      0.79      0.79      2612


Confusion Matrix:
 [[367   9  98  56   0]
 [  4 405  68   1  49]
 [ 67  63 330  10   1]
 [ 69   1  11 457   0]
 [  0  37   4   0 505]]
```


Top 10 Feature Importances
BikeRentals
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
hour
humidity
wind_speed
temperature_feels
temperature
is_weekend
weather_code_1
season_3
weather_code_2
weather_code_7
Feature