Task: Based on rented bike count, the hour of the day, the day's temperature, humidity,

wind speed, rainfall, holidays, and many other factors, build a model to predict the bike

count required each hour for a stable supply of rental bikes.

• Load the dataset

• Check for null values in any columns and handle the missing values

• Convert Date columns to Date format and extract day, month, day of week, and

weekdays/weekend from date column

• Check correlation of features using heatmap

• Plot the distribution plot of Rented Bike Count

• Plot the histogram of all numerical features

• Plot the box plot of Rented Bike Count against all the categorical features (Hint: Categorical

features on X-axis and Rented Bike Count on Y-axis)

Plot the Seaborn catplot of Rented Bike Count against features like Hour, Holiday, Rainfall

(mm), Snowfall (cm), weekdays, weekend, and give your inferences.

• Encode the categorical features into numerical features.

(Hint: use get\_dummies())

• Identify the target variable and split the dataset into train and test with a ratio of 80:20

and random state 1

• Perform Standard Scaling of the train dataset.

• Perform Linear Regression, Lasso Regression, and Ridge Regression to predict the bike

count required each hour and compare the results.

**Remaining Tasks**

* **Date Conversion and Feature Extraction:**
  + Convert the 'Date' column to a datetime format.
  + Extract the day, month, and day of the week from the 'Date' column.
* **Data Splitting:**
  + Identify the target variable (Rented Bike Count).
  + Split the dataset into training (80%) and testing (20%) sets.
* **Standard Scaling:**
  + Perform standard scaling on the training data.
* **Regression Modeling:**
  + Build and compare the results of the following regression models:
    - Linear Regression
    - Lasso Regression
    - Ridge Regression

**Project Task Status**

Here is a breakdown of what has been done and what is left for the project, organized by each task.

**Task 1: Load and Check the Dataset**

* **Requirement**: Load the dataset and check for null values1111.
* **Status**: ✅ **Completed**
* **Details**: You successfully loaded the FloridaBikeRentals.csv file into a pandas DataFrame. You also checked for missing values and duplicates, confirming that the dataset was clean in that regard.

**Task 2: Date Conversion and Feature Extraction**

* **Requirement**: Convert the 'Date' column to a datetime format and extract the day, month, and day of the week2.
* **Status**: ❌ **Not Started**
* **Details**: The 'Date' column remains as an object type in your notebook. There is no code for converting it or extracting additional features like the day or month.

**Task 3: Data Visualization**

* **Requirement**: Create a heatmap for feature correlation, a distribution plot for Rented Bike Count, a histogram for all numerical features, a box plot of Rented Bike Count against categorical features, and a Seaborn catplot for Rented Bike Count against various features3333.
* **Status**: partial **Partially Completed**
* **Details**: You have successfully created a **heatmap** and **box plots** for Rented Bike Count and Temperature(°C). However, the **distribution plot**, **histograms**, and **catplot** are missing.

**Task 4: Categorical Feature Encoding**

* **Requirement**: Encode categorical features into numerical ones, with a hint to use get\_dummies()4444.
* **Status**: ✅ **Completed**
* **Details**: You correctly used pd.get\_dummies() to one-hot encode the 'Seasons', 'Holiday', and 'Functioning Day' columns and saved the result to a new CSV file.

**Task 5: Data Splitting**

* **Requirement**: Identify the target variable and split the dataset into an 80:20 train-test ratio with a random state of 15555.
* **Status**: ❌ **Not Started**
* **Details**: The notebook does not yet contain code for splitting the data into training and testing sets.

**Task 6: Standard Scaling**

* **Requirement**: Perform standard scaling on the training dataset6666.
* **Status**: ❌ **Not Started**
* **Details**: Since the data has not been split, there is no implementation of standard scaling.

**Task 7: Regression Modeling**

* **Requirement**: Perform and compare Linear Regression, Lasso Regression, and Ridge Regression to predict the hourly bike count7777.
* **Status**: ❌ **Not Started**
* **Details**: The notebook's analysis currently stops after data exploration and does not include any of the required regression models.

**How your notebook stacks up against the Session-5 “Bike-Rental” checklist**

| **#** | **Requirement (from the spec)** | **Status in kourosh sohrabi Incremental Capstone.ipynb** | **Evidence / tips** |
| --- | --- | --- | --- |
| 1 | **Load the dataset** | **✓ Done** (pd.read\_csv('FloridaBikeRentals.csv')) | cell 2 |
| 2 | **Check & impute missing values** | **✗ Missing** – no isnull().sum() / fillna logic found | add a quick null‐audit & decide on strategy |
| 3 | **Convert Date → datetime & derive day, month, dow, weekend** | **✗ Missing** | after df['Date']=pd.to\_datetime(...) use dt.\* accessors |
| 4 | **Correlation heatmap** | **✓ Done** (sns.heatmap) | cell 9 |
| 5 | **Distribution plot of Rented Bike Count** | **✗ Missing** – only a box-plot is drawn | one-liner: sns.histplot(df['Rented Bike Count'], kde=True) |
| 6 | **Histogram of all numerical features** | **✗ Missing** | loop over df.select\_dtypes('number') |
| 7 | **Box-plot of Rented Bike Count vs every categorical feature** | **∼ Partially** – single box-plot of the target only | iterate through each categorical column and call sns.boxplot(x=col, y='Rented Bike Count', data=df) |
| 8 | **Seaborn catplot against Hour, Holiday, Rainfall, Snowfall, weekday/weekend + inferences** | **✗ Missing** | e.g. sns.catplot(data=df, x='Hour', y='Rented Bike Count', kind='box') etc.; add a markdown cell with takeaways |
| 9 | **One-hot encode categoricals with get\_dummies** | **✓ Done** for Seasons, Holiday, Functioning Day | cell 7 – consider encoding weekday/weekend too |
| 10 | **Train/test split 80 : 20 (random\_state=1)** | **✗ Missing** | train\_test\_split(X, y, test\_size=0.2, random\_state=1) |
| 11 | **Standard Scaler on train features** | **✗ Missing** | wrap in a Pipeline to avoid leakage |
| 12 | **Fit & compare Linear, Ridge, Lasso regression** | **✗ Missing** | instantiate, fit on scaled train data, evaluate R2R^2/RMSE on hold-out |

The full checklist is in the project brief.

**Key gaps & next steps**

1. **Data quality**
   * Add a null-value audit (df.isnull().sum()) and imputation or row-drop logic.
   * Parse the Date column once, then create:**df['day'], month, dayofweek, weekend**.
2. **Exploratory visuals**
   * Draw a **histogram** or **dist-plot** of the target.
   * Loop over numerical columns for histograms.
   * Loop over categorical columns for box-plots against the target.
   * Build the requested **catplots** and drop a short markdown cell with insights (e.g., “Rentals spike on weekends …”).
3. **Modelling pipeline**
4. from sklearn.model\_selection import train\_test\_split
5. from sklearn.preprocessing import StandardScaler
6. from sklearn.pipeline import make\_pipeline
7. from sklearn.linear\_model import LinearRegression, Ridge, Lasso
8. X = df\_encoded.drop('Rented Bike Count', axis=1)
9. y = df\_encoded['Rented Bike Count']
10. X\_train, X\_test, y\_train, y\_test = train\_test\_split(
11. X, y, test\_size=0.20, random\_state=1)
12. models = {
13. 'Linear': LinearRegression(),
14. 'Ridge' : Ridge(alpha=1.0),
15. 'Lasso' : Lasso(alpha=0.01, max\_iter=10000)
16. }
17. results = {}
18. for name, mdl in models.items():
19. pipe = make\_pipeline(StandardScaler(with\_mean=False), mdl)
20. pipe.fit(X\_train, y\_train)
21. results[name] = pipe.score(X\_test, y\_test) # R²
22. print(results)
23. **Documentation**
    * Use markdown headings before each major block (“### Handle missing values”, “### Feature engineering …”) so reviewers can trace requirement → code quickly.

Implementing the above will close all unmet items and produce a notebook fully aligned with the brief. Need more code samples or explanations for any specific step? Let me know!