CS7CS4/CSU44061 Machine Learning Devang Kankaria
23335884

Final Assignment 2023-24

Part A

Data Preprocessing

1. *Data Cleaning*:
   - Post noticing the discrepancy in the column names under the range of all the data files, the column names were rectified to maintain uniformity in the combined dataset (explained later).
   - Filling the missing data values: The missing data inconsistencies in all the individual datasets were identified and corrected by replacing the cells with 0 as per its unavailability.
   - For classification of duplicate column names, all the datasets were scanned and verified if there was any occurrence of duplication or recurrence.
2. *Data Integration*:
   - Post Data Cleaning, all the datasets were merged to create a comprehensive and coherent set for performing the exploratory data analysis along with implementation of Machine Learning algorithms for assessing the impact of the pandemic on the city-bike usage for the pandemic and post-pandemic period.
3. *Data Reduction*:
   - Redundant features were excluded to bring attention to the features which potentially held vital information for "bikes taken".
   - Instead of executing explicit data reduction practice, I went ahead with creation of an independent dataset for performing the adequate requirements listed without any unusual interruptions from the current dataset's inadvertent constraints and redundant columns.
4. *Data Transformation*:
   - Column TIME was trimmed down from the format '%Y-%m-%d' to '%Y-%m-%d' to extract the date section of the original datetime string. This makes it convenient for the viewer for exploratory data analysis on a 3-day basis, thereby rendering a more comprehensive plot and underlying patterns to grasp the nature of the dataset's run over the span of August 1, 2018 to December 30, 2023.

Feature Engineering

is a definitive step in the pathway of a predictive model's development. It comprises translating the collection of raw data to fitting and suitable features which are desirable for a detailed illustration of the tasks at hand to the predictive models. This accelerates the model's accuracy & its performance on unseen data.

- Data Transformation: 'TIME' column denoting the date (post performing Data Transformation) is converted to a format for my predictive model. In order to do so, I have undertaken the approach of converting the regular dates to ordinal numbers, which performs counting the total number of days from a particular starting point. The pandemic data has been opted to be taken from March 1, 2020, so that I have 3 fair & circa equal instalments of data for each stretch of time:
- Management of Pandemic Periods: The data has been broken up into 3 partitions to be answerable to the assessment of the pandemic inflictions. Hence, the pandemic period was studied thoroughly to subsequently predict the nature of the Bike Usage had the pandemic hadn't happened. This has been done for both Pandemic and Post Pandemic in compliance with the requirements.

Pre-Pandemic	August 1, 2018 - February 28, 2020
Pandemic	March 1, 2020 - July 31, 2021
Post Pandemic	August 1, 2021 - December 30, 2023

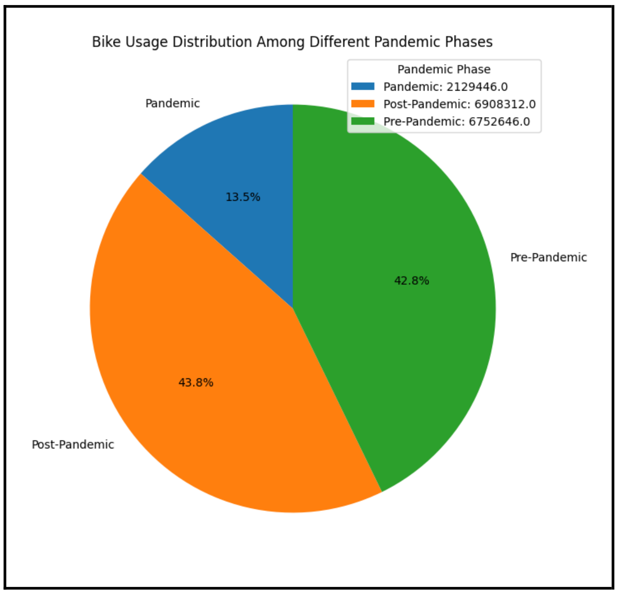
- Predictions: I have concentrated upon the pre-pandemic model to assess the predictions of the bike usage during the period of pandemic and post-pandemic as a serial measure of how the bike usage would have panned out had the pandemic hadn't happened.
- Introduction of new columns: new columns were embedded in the original dataset:

net bikes rented	to compute the values of the total number of bikes rented
change in available bikes	the evolution of the change in the nature of bike availability
Period	the correct assignment of the phase of the time

Bike Usage Distribution Among Different Pandemic Phases

```
bike_usage_by_period = df.groupby('Period')['bikes taken'].sum()
```

- Pre-pandemic:
  - Total bike usage = 6752646
- Pandemic:
  - Observed a visible decline to the count of: 2129446 i.e. a 68.4% diminution.
- Post-pandemic:
  - There was an observable restoration in the bike usage with an increase of ¬2.3% with keeping the pre-pandemic phase as a point of reference.



Representation of Average Available Bikes during all the 3 phases: Pre-Pandemic | Pandemic | Post-Pandemic

df	• The primary dataset holds the raw data (dataset before performing Data Preprocessing. • Shape: (42137028, 15)
altered_df	• Data post Data Preprocessing, with the shape of (660,2) • The two columns being 'TIME' and 'bikes taken '

I had initially calculated 'net bikes rented' by computing the difference between Total Bike Stands and Available Bike Stands:

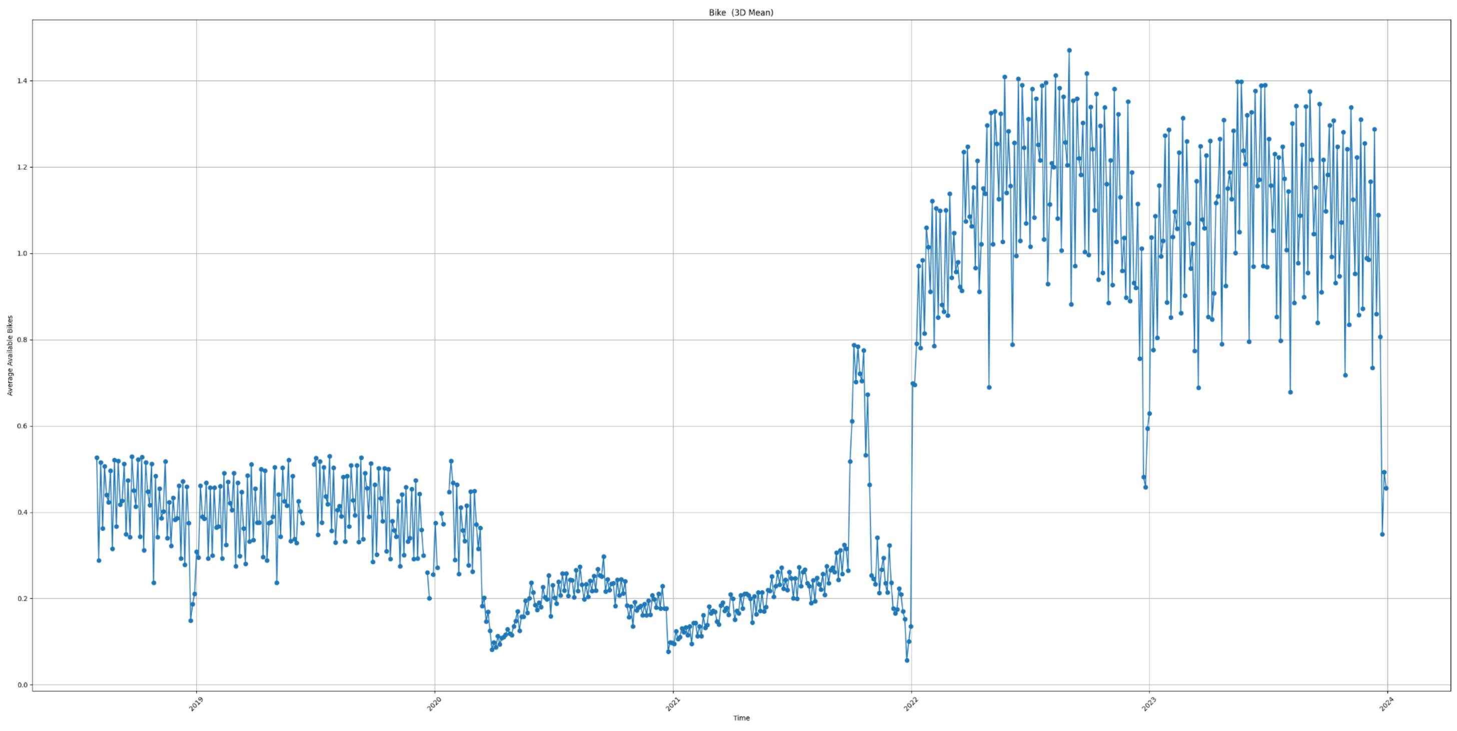
```
df['net bikes rented'] = df['BIKE STANDS'] - df['AVAILABLE BIKE STANDS']
```

Now, adjacent differences are computed to calculate the Change in Available Bikes from one record to the next.

```
df['change in available bikes'] = df.groupby('STATION ID')['AVAILABLE BIKES'].diff()
```

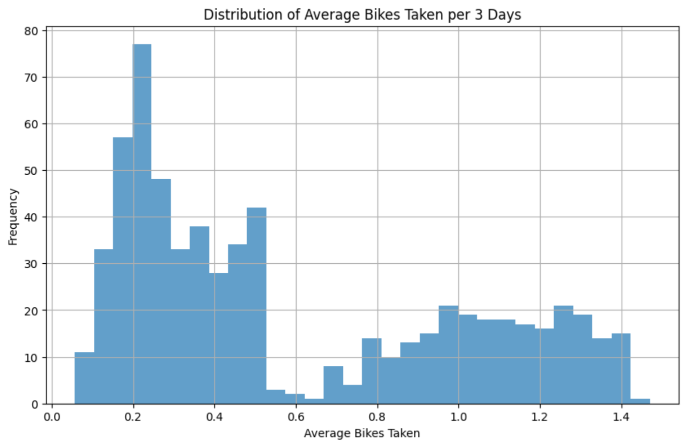
The absolute value for 'Change in Available Bikes' will be the 'bikes taken' values.

Pre-Pandemic Pandemic Post-Pandemic



Distribution of Average Bikes Taken on a 72 hour basis:

- depicts the delineation of a histogram plot
- Frequency of Average Bikes Taken over the span of 2018-2023.
- Showcases the Crest and Nadir of the average bikes taken value.
- This helped me describing the Visual Distribution, identifying the Central Tendency, assessing skewness, and understanding the spread and dispersion in my Dublinbikes merged dataset.



Machine learning methodology

Post handling the missing values, I went ahead with the modelling procedure.
Steps:

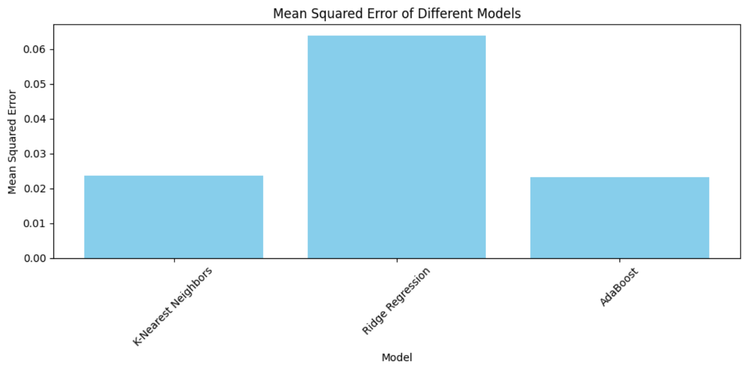
- X & y are dedicatedly defined and Training and Testing data are outlined using train_test_split with 20% test size.
- This Xtrain and train is undergone feature scaling with StandardScaler()

- The models I employed here are:
  - ➢ K-Nearest Neighbors
  - ➢ Ridge Regression
  - ➢ AdaBoost
- Training and prediction was executed taking each model into account.
- Now. MSE (Mean-Squared Error) was computed and the scores were utilised to facilitate easy access and clear judgement of picking the right model for manifesting the predictions.
- The MSE scores are as follows:
-

Model	**MSE**
K-Nearest Neighbors	0.3256
Ridge Regression	0.3394
AdaBoost	0.3398

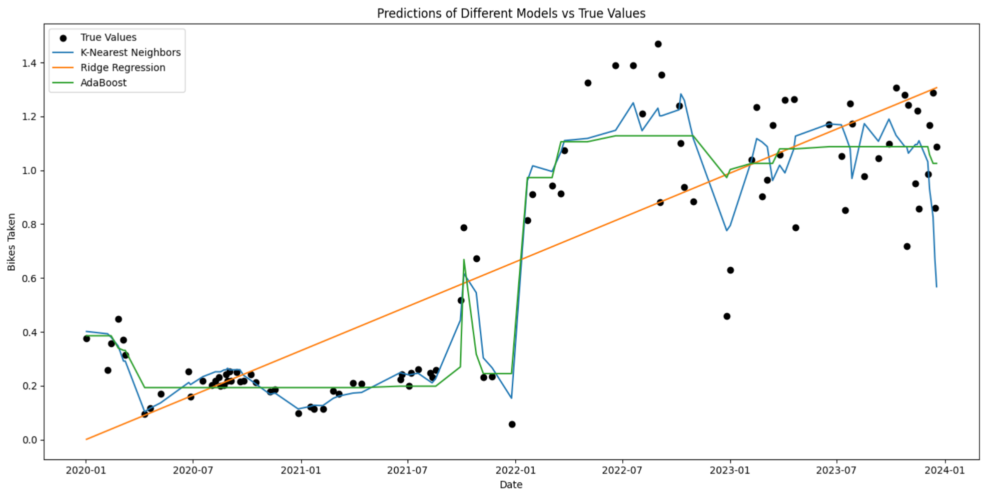
The plot here gives a lucid picture of the MSE values of the models that I utilised for training.

Apparently, there happened to be a close squeeze between K-Nearest Neighbor and AdaBoost model, with Adaboost edging K-Nearest Neighbor out by the barest of all margins.



Prediction of Various Models used as opposed to the True Values

From the plot below, we can get an accurate understanding of the prediction performance of all the 3 models. This serves as a testament to the model's accomplishment in manifesting the predictions.



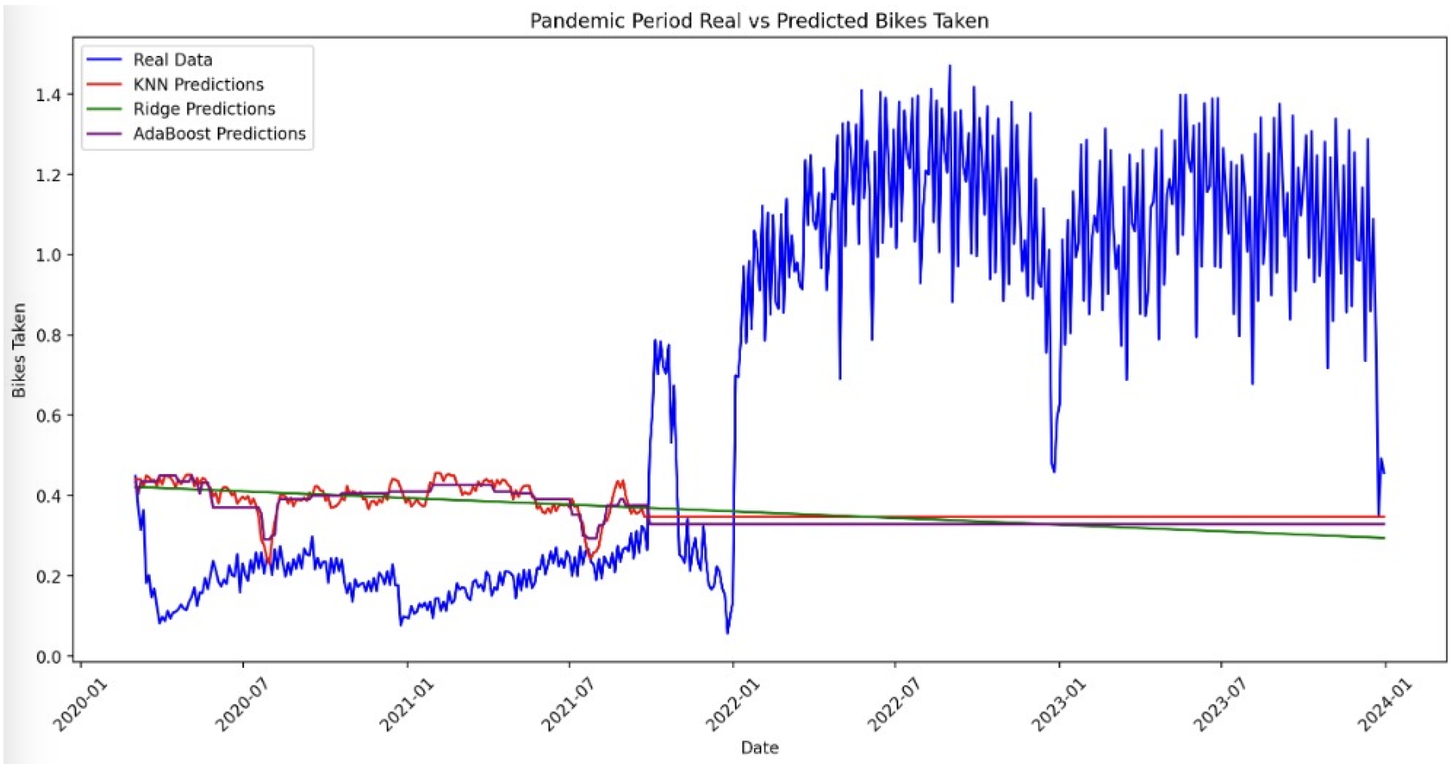
Devang Kankaria
23335884

Predictions, had the pandemic hadn't happened

Goals:

1. To assess the impact of the pandemic on the city-bike usage for the pandemic period.
2. To assess the impact of the pandemic on the city-bike usage for the post-pandemic period.

Pandemic Post-pandemic



Here, I plotted the predicted data generated, had the pandemic hadn't happened) from all of my 3 models (K-Nearest Neighbors, Ridge Regression, and AdaBoost) trained on Pre-pandemic data. The plot spans from the start of the Pandemic (March 1, 2020 in my case) till the end of Post-pandemic (December 30, 2023). This is a comprehensive representation of the predicted values from my models, assessing and delineating the bikes taken if pandemic wasn't a concern.

KNN MSE for Pandemic Period	0.3256
Ridge MSE for Pandemic Period	0.3394
AdaBoost MSE for Pandemic Period	0.3398

As could be gleaned from the plot above, the predicted bikes taken mean values from my models speak that the pattern remains in the column of 0.2 to 0.5 till early January and then follow the trend of a straight line with slope zero degrees (parallel to x-axis), which can be juxtaposed with the True values of the mean bikes taken.

Part B

1) What is a ROC curve? How can it be used to evaluate the performance of a classifier compared with a baseline classifier? Why would you use an ROC curve instead of a classification accuracy metric?

- *What is a ROC curve?*
    - ➢ ROC (Receiver Operating Characteristic) curve: serves as a graphical/visual representation of the performance of a classifier.
    - ➢ The Trade-Off between TPR [True Positive Rate] & FPR [False Positive Rate] is exemplified between disparate values of thresholds.
- *How can it be used to evaluate the performance of a classifier compared with a baseline classifier?*
    - ➢ A better classifier is evident from its curve geared more towards the top-left corner on the plot of TPR vs FPR, a symbolism of a classifier with Higher True Positive Rates and Lower False Positive Rates. Baseline Classifier has its ROC Curve falling around the diagonal line on the TPR vs FPR plot (as is evident from the aforementioned plot).
    - ➢ AUC for Baseline Classifier hovers on 0.5, depicting its mediocrity and failure to classify the data points immaculately, signifying feeble ability to classify correctly between classes.
    - ➢ If the plot lies below the diagonal line depicted in the TPR vs FPR plot, it signifies that the classifier is an underperforming classifier.
- *Why would you use an ROC curve instead of a classification accuracy metric?*
    - ➢ An ROC curve undertakes a broad assortment of classification thresholds. By running the gamut on these diverse values, it renders valuable insights beyond a single accuracy metric.
    - ➢ Highly effective & especially relevant when situations pertaining to imbalanced datasets, having misleading accuracy.
    - ➢ Enables a more nuanced understanding of a classifier's performance by revealing its ability to differentiate between classes.

ROC curves are formidable means for computing the performance of a classifier, with remarkable performance in instances where a levelheaded perspective of TPR & FPR is paramount. Not only do they render a graphical representation & an all-around perspective, they also help in devising an informed decision-making process.

---

2) Give two examples of situations where a linear regression would give inaccurate predictions. Explain your reasoning and what possible solutions you would adopt in each situation.

Examples	Situation	Conviction	Solution
Outliers in the dataset	When it comes to outliers, Linear Regression is sensitive to them, them being the data points that are outrageously disparate as compared to the rest of the dataset. This can lead to false construction of the regression line, leading to fallacious predictions.	In case of dataset's extreme values which don't represent the common trend but have a strong impact on the regression line, the model, for new data points, might give fallacious predictions.	Outlier identification is necessary before the linear regression model is fit. This could also entail removal of outliers if they are erroneous or irregular observations or aberrations.

Non-linear Relationships	Linear Regression expects a linear relationship between dependent and independent variables. Linear regression could lead to incorrectness in the results.	Let's assume we're predicting the price of a house based on its features, and the actual relation between the features and the price shadows a quadratic pattern, then a linear model would fail to absorb the essence of this intricacy precisely.	A polynomial regression model or other non-linear regression methods can turn out to be a good proposal as these models would absorb more convoluted relationships by bringing in higher-order terms or implementing non-linear shifts to the features.

---

3) *The term 'kernel' has different meanings in SVM and CNN models. Explain the two different meanings. Discuss why and when the use of SVM kernels and CNN kernels is useful, as well as mentioning different types of kernels.*

Kernel	Meaning	When & Why?	Types
SVM Kernel	In SVM, a kernel moves the training data into relatively-higher dimensions. SVM focuses on crafting a hyperplane that best segregates the data points of distinct classes. Kernel's use permits SVM to function in a higher-dimensional space without specifically computing the new feature vectors. SVMs work by moving the data into relatively high dimensional space and finding a relatively high SVC that can effectively classify the observations.	1) Non Linearity: SVM kernels come handy when the relation between the input features and the output is of a non-linear fashion. Kernels permit SVM to track intricate decision boundaries that linear hyperplane fails to achieve. 2) Kernels aid in implicitly modification of the input features into a higher-dimensional space, thereby facilitating the detection of a hyperplane for separating purposes.	1) Linear Kernel: exercised for data which is linearly distinct. 2) Polynomial Kernel: which has a parameter d = degree of the polynomial. Befitting for data with polynomial relationships. 3) Radial Basis Function (RBF) Kernel(aka: Radial Kernel): Finds SVC in infinite dimensions
CNN Kernel	In CNNs, a kernel(aka filter) is a small matrix utilised in the convolutional layer to execute the convolution tasks. Dot product is taken between input and the filter at each location of the kernel's traversal, thus helping in collecting internal features and patterns. We can say that the filter is Convolved with the input, hence Convolutional NN.	1) CNN kernels are developed to collect the native features, assisting the network for pattern identification & object detection in various portions of input image. 2) The network can perform pattern identifications regardless of their placement in the image. 3) CNNs are constructed as computationally effectual and efficient as the utilisation of shared weights in the kernels decreases the number of parameters.	1) Edge Detection Kernel: To identify edges in the images. (such as Prewitt, Roberts, Sobel Kernel) 2) Gaussian Blur Kernels: utilised for noise reduction and image blurring. 3) Identity Kernel: $\begin{array}{\|c\|c\|c\|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$

4) In k-fold cross-validation, a dataset is resampled multiple times. What is the idea behind this resampling i.e. why does resampling allow us to evaluate the generalisation performance of a machine learning model. Give a small example to illustrate. Discuss when it is and it is not appropriate to use k-fold cross-validation.

- *What is the idea behind this resampling i.e. why does resampling allow us to evaluate the generalisation performance of a machine learning model?*
    - ➢ The underlying idea is to achieve a far more powerful evaluation of the model's performance by taking dataset's different batches for training and testing, to make sure that the performance of the model is not highly swayed by the explicit data points in a single split.
    - ➢ K-fold Cross Validation is a technique exercised for the evaluation of machine learning model's performance by splitting the data into k partitions.
    - ➢ The purpose of this technique is to decrease overfitting by utilising all the data points for training & testing the model on different sets of data.

Example:
For example, we have a dataset of 100 samples, & perform 5-fold Cross-Validation. So, the dataset is split into 5 folds, each split having 20 samples. Now, we're training the model 5 times, using a different fold as the test set in each iteration.

- ➢ Iteration 1: Training data on Folds 2-5, Testing data on Fold 1
- ➢ Iteration 2: Training data on Folds 1, 3-5, Testing data on Fold 2
- ➢ Iteration 3: Training data on Folds 1-2, 4-5, Testing data on Fold 3
- ➢ Iteration 4: Training data on Folds 1-3, 5, Testing data on Fold 4
- ➢ Iteration 5: Training data on Folds 1-4, Testing data on Fold 5

The conclusive performance metric is further computed on the basis of performing the mean of the distinct metrics captured during every iteration.
Hence, this is how resampling allows us to evaluate the generalisation performance of our machine learning model.

<u>Appropriateness of k-fold cross-validation</u>: K-fold cross-validation is apt in case of finite collection of data, and when we desire to maximise the utilisation of available samples for both training and testing purposes, resulting in a reliable and stable assessment of the model's performance.

<u>Not Appropriate of k-fold cross-validation</u>: K-fold cross-validation could be computationally expensive in case of colossal datasets. Hence, a simple train/test split might suffice. Moreover, for time-series data, where the samples' order is paramount, k-fold cross-validation might be unbecoming. Instead, time-series-specific techniques like time series split are more apt.

APPENDIX

```
import pandas as pd
import matplotlib.pyplot as plt

# Load your dataset
df = pd.read_csv('\combined_cleaned_data.csv', parse_dates=['TIME'])


pandemic_start = pd.Timestamp('2020-03-01')
pandemic_end = pd.Timestamp('2021-03-01')

def categorize_period(row):
    if row < pandemic_start:
        return 'Pre-Pandemic'
    elif pandemic_start <= row <= pandemic_end:
        return 'Pandemic'
    else:
        return 'Post-Pandemic'

df['Period'] = df['TIME'].apply(categorize_period)

bike_usage_by_period = df.groupby('Period')['bikes taken'].sum()

total_bikes_used=bike_usage_by_period.sum()
plt.figure(figsize=(8, 8))
bike_usage_by_period.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title('Bike Usage Distribution Among Different Pandemic Phases')
plt.ylabel('')

legend_labels = [f'{phase}: {count}' for phase, count in zip(bike_usage_by_period.index, bike_usage_by_period)]
plt.legend(legend_labels + [f'Total: {total_bikes_used}'], title='Pandemic Phase', loc='upper right')

plt.show()

df.head()

import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('\combined_cleaned_data.csv', parse_dates=['TIME'])
df['TIME'] = pd.to_datetime(df['TIME'])
df['TIME']=df['TIME'].dt.strftime('%Y-%m-%d')
df['TIME'] = pd.to_datetime(df['TIME'])
df.set_index('TIME',inplace=True)
altered_df=df['bikes taken'].resample('3D').mean()
```

```
altered_df.head()

altered_df.shape

altered_df.to_csv('altered_data.csv',index=True)

plt.figure(figsize=(30, 15))
plt.plot(altered_df.index, altered_df.values, marker='o', linestyle='-')
plt.title('Bike  (3D Mean)')
plt.xlabel('Time')
plt.ylabel('Average Available Bikes')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

!pip install tqdm

import pandas as pd
from tqdm import tqdm

file_path = 'altered_data.csv'
df = pd.read_csv(file_path, encoding='ascii')

df_head = df.head()
print(df_head)

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
df['bikes taken'].plot(kind='hist', bins=30, alpha=0.7)
plt.title('Distribution of Average Bikes Taken per 3 Days')
plt.xlabel('Average Bikes Taken')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

!pip install scikit-learn

df['TIME']=pd.to_datetime(df['TIME'])
df.sort_values('TIME',inplace=True)

pandemic_start = pd.to_datetime('2020-01-01')
df_pandemic = df[df['TIME'] >= pandemic_start]

df_pandemic.isnull().sum()

df_pandemic.dropna(subset=['bikes taken'], inplace=True)
df_pandemic.isnull().sum()

from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from tqdm import tqdm
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Ridge
from sklearn.ensemble import AdaBoostRegressor

X = df_pandemic[['TIME']].apply(lambda x: x.map(datetime.toordinal)) # Convert dates to ordinal
y = df_pandemic['bikes taken']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

models = {
    'K-Nearest Neighbors': KNeighborsRegressor(),
    'Ridge Regression': Ridge(),
    'AdaBoost': AdaBoostRegressor()
}

predictions = {}

knn_model = models['K-Nearest Neighbors']
knn_model.fit(X_train_scaled, y_train)
predictions['K-Nearest Neighbors'] = knn_model.predict(X_test_scaled)

ridge_model = models['Ridge Regression']
ridge_model.fit(X_train_scaled, y_train)
predictions['Ridge Regression'] = ridge_model.predict(X_test_scaled)

adaboost_model = models['AdaBoost']
adaboost_model.fit(X_train_scaled, y_train)
predictions['AdaBoost'] = adaboost_model.predict(X_test_scaled)

mse_scores = {
    'K-Nearest Neighbors': mean_squared_error(y_test, predictions['K-Nearest Neighbors']),
    'Ridge Regression': mean_squared_error(y_test, predictions['Ridge Regression']),
    'AdaBoost': mean_squared_error(y_test, predictions['AdaBoost'])
}

print(mse_scores)

import matplotlib.pyplot as plt

model_names = list(mse_scores.keys())
mse_values = list(mse_scores.values())

plt.figure(figsize=(10, 5))
plt.bar(model_names, mse_values, color='skyblue')
plt.title('Mean Squared Error of Different Models')
plt.ylabel('Mean Squared Error')
plt.xlabel('Model')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

!pip install numpy

from datetime import datetime
import numpy as np

X_test_dates = scaler.inverse_transform(X_test_scaled)

X_test_dates = [datetime.fromordinal(int(date)) for date in X_test_dates.flatten()]

sorted_indices = np.argsort(X_test_dates)
X_test_dates_sorted = np.array(X_test_dates)[sorted_indices]

plt.figure(figsize=(14, 7))
plt.scatter(X_test_dates_sorted, y_test.iloc[sorted_indices], color='black', label='True Values')

for name, y_pred in predictions.items():
    plt.plot(X_test_dates_sorted, y_pred[sorted_indices], label=name)

plt.title('Predictions of Different Models vs True Values')
plt.xlabel('Date')
plt.ylabel('Bikes Taken')
plt.legend()
plt.tight_layout()
plt.show()

pandemic_data = altered_df.loc[altered_df.index >= '2020-03-01']

pandemic_data['days_since_start'] = (pandemic_data.index - pandemic_data.index[0]).days
```

```
X_pandemic = pandemic_data['days_since_start'].values.reshape(-1, 1)
y_pandemic = pandemic_data['bikes taken'].values

knn_predictions_pandemic = knn_model.predict(X_pandemic)
ridge_predictions_pandemic = ridge_model.predict(X_pandemic)
ada_predictions_pandemic = ada_model.predict(X_pandemic)

plt.figure(figsize=(15, 7))
plt.plot(pandemic_data.index, y_pandemic, label='Real Data', color='blue', linestyle='-')
plt.plot(pandemic_data.index, knn_predictions_pandemic, label='KNN Predictions', color='red', linestyle='-')
plt.plot(pandemic_data.index, ridge_predictions_pandemic, label='Ridge Predictions', color='green', linestyle='-')
plt.plot(pandemic_data.index, ada_predictions_pandemic, label='AdaBoost Predictions', color='purple', linestyle='-')
plt.title('Pandemic Period Real vs Predicted Bikes Taken')
plt.xlabel('Date')
plt.ylabel('Bikes Taken')
plt.legend()
plt.show()
```