# Part - I

## Requirements

In order to address the objectives, set by Dublin City Council and meet the goals outlined by FUTURE-DATA, my assignment focuses on evaluating the impact of the COVID-19 pandemic on the city-bike usage in two distinct periods: during the pandemic and in the post-pandemic phase. To accomplish this, I must employ a multifaceted approach that combines descriptive statistics with machine learning techniques. The Dublinbikes dataset, obtained from https://data.gov.ie/dataset/dublinbikes-api, is organized quarterly and monthly, necessitating the concatenation of various portions. My primary task involves utilizing machine learning model to estimate city-bike usage under hypothetical conditions where the pandemic did not occur. I consider the creation of a derived "bike usage" feature crucial for the analyses, distinguishing it from the original features related to bike and bike stand availability. Throughout the report, my choices and methodologies should be meticulously justified, ensuring a clear and comprehensive discussion of the results. I should present my findings with concise and visually informative figures.

## Data Pre-Processing and Feature Engineering

1. **Downloaded the Data CSV File from Dublin Bikes:**

   I started the process by obtaining the data CSV file directly from Dublin Bikes. The files likely contain comprehensive information about bike stations, stands, and bike availability.

2. **Split Data Files into Three Timeline Folders: Pre-Pandemic, Pandemic, and Post Pandemic:**

   I organized the data into three separate timeline folders, namely pre-pandemic, pandemic, and post-pandemic. This categorization was done based on timestamps, reflecting different periods of interest.

   My pandemic period timeline is based majorly on the major lockdown periods as they have the biggest impact on commute.

3. **Remove Unnecessary Columns and Rename Columns:**

- To streamline the dataset, I identified and removed unnecessary columns such as 'NAME', 'STATUS', 'ADDRESS', 'LATITUDE', 'LONGITUDE', and 'LAST UPDATED'.
- I also ensured consistency by renaming columns, for instance, 'BIKE STANDS' to 'BIKE_STANDS', 'AVAILABLE BIKE STANDS' to 'AVAILABLE_BIKE_STANDS', 'STATION ID' to 'STATION_ID', and 'AVAILABLE BIKES' to 'AVAILABLE_BIKES'.

4. **Round the Timestamp to Hour:**

   For each data point, I rounded the timestamp to the nearest hour. This step was essential for creating a more aggregated and manageable time granularity.

   ```
   df['TIME'] = df['TIME'].dt.floor('H')
   ```

5. **Group by Date and Time, and Aggregate the Data:**

   I grouped the information by station ID and timestamp. Subsequently, I aggregated relevant bike metrics (BIKE_STANDS, AVAILABLE_BIKE_STANDS, AVAILABLE_BIKES) by calculating their mean values.

```
df_modpostpandemic = df.groupby(['STATION_ID', 'TIME']).agg({
  'BIKE_STANDS': 'mean',
  'AVAILABLE_BIKE_STANDS': 'mean',
  'AVAILABLE_BIKES': 'mean'
```

6. **Calculate Bike Usage:**

   I implemented a function called **calculate_bike_usage** to determine the net change in available bikes at each station and timestamp. The function appropriately handled missing values and adjusted for negative values.

```
def calculate_bike_usage(df):
  # Calculate the net change in bikes
  df['BIKE_USAGE'] = df['AVAILABLE_BIKES'] - df['AVAILABLE_BIKES'].shift(1)
```

   This line calculates the net change in the number of available bikes by subtracting the previous day's available bikes from the current day's available bikes. The **shift(1)** function shifts the values in the 'AVAILABLE_BIKES' column by one day, aligning the subtraction operation.

```
  # Handle missing values in the first row
  df['BIKE_USAGE'] = df['BIKE_USAGE'].fillna(0)
```

   In the first row, where there is no previous day's value to subtract, the subtraction would result in a missing value (NaN). This line fills those missing values with 0, assuming no change in bike usage on the first day.

```
  # Adjust for negative values
  df['BIKE_USAGE'] = df['BIKE_USAGE'].apply(lambda x: max(0, x))
```

   This line ensures that the 'BIKE_USAGE' values are non-negative. Any negative values resulting from the subtraction (indicating an increase in available bikes) are replaced with 0, as negative bike usage doesn't make sense in this context.

7. **Calculate Daily Sum by Station ID:**

   Using the calculated bike usage, I aggregated the data by station ID and day to obtain the daily sum of bike usage per station.

```
df_daily = df.groupby(['STATION_ID', pd.Grouper(key='TIME',
freq='D')])['BIKE_USAGE'].sum().reset_index()
```

8. **Calculate Daily Sum of Citywide Usage:**

   I computed the daily sum of citywide bike usage by aggregating the data based on the timestamp.

```
df_city_daily = df.groupby('TIME')['BIKE_USAGE'].sum().reset_index()
```

9. **Combine Three Data Files into One Single CSV File:**

   After processing the data for each timeline, I consolidated the results into a single CSV file. This

comprehensive file provides an overview of citywide daily bike usage.

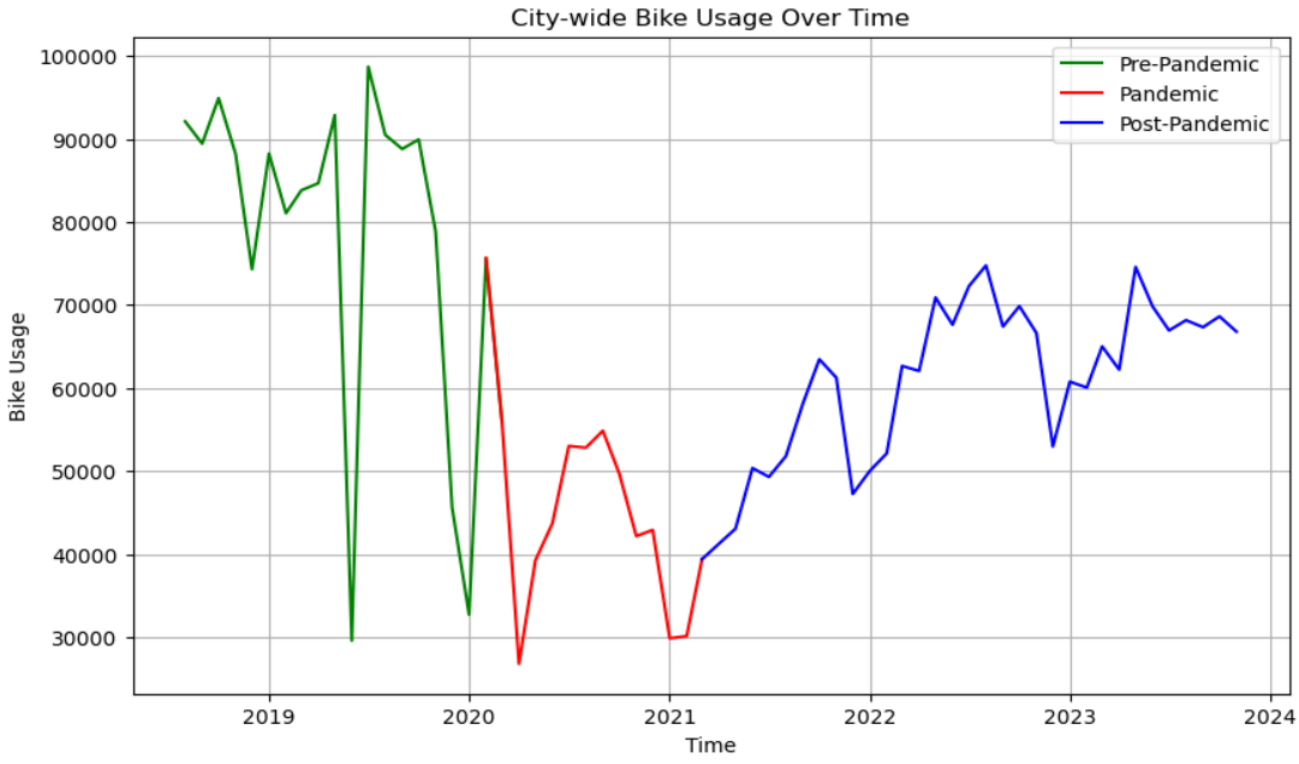
10. **Convert Daily Data to Monthly for Better Visualization:**

To enhance visualization, I converted the daily data to a monthly format. This likely involved grouping the data by month and aggregating the daily sums.

```
# Resample the data by month and sum the 'BIKE_USAGE' column

df_monthly = df.resample('M').sum()

# Reset the index and convert the time back to YYYY-MM format

df_monthly.reset_index(inplace=True)

df_monthly['TIME'] = df_monthly['TIME'].dt.to_period("M").astype(str)
```

These preprocessing steps were crucial in preparing the data for subsequent analysis and visualization, offering valuable insights into bike usage patterns over different timelines.

**Graph 1**



## Machine Learning Methodology and Evaluation

**Linear Regression:**

Linear Regression is a simple yet powerful regression model that establishes a linear relationship between the input features and the target variable. It assumes a linear combination of features to predict the continuous target variable. In the context of bike usage prediction, linear regression attempts to find the best-fit line through the data, minimizing the sum of squared differences between predicted and actual bike usage.

**Ridge Regression:**

Ridge Regression is an extension of linear regression that addresses the issue of multicollinearity in the feature space. It introduces a regularization term to the linear regression cost function, penalizing large coefficients. This regularization helps prevent overfitting and improves the model's generalization performance. Ridge Regression is particularly useful when dealing with datasets with highly correlated features.

**k-Nearest Neighbours (KNN):**

k-Nearest Neighbours is a non-parametric and instance-based learning algorithm used for both classification and regression tasks. In the context of bike usage prediction, KNN calculates the bike usage for a specific day by averaging the bike usage of its k nearest neighbours in the feature space. The choice of k influences the model's sensitivity to local fluctuations and impacts its generalization ability.

**Long Short-Term Memory (LSTM):**

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem, which is common in traditional RNNs. LSTMs are particularly well-suited for sequence prediction tasks, making them a powerful choice for time-series data such as bike usage prediction.

**Support Vector Machine (SVM):**

Support Vector Machines are powerful supervised learning models that can be applied to both classification and regression problems. In regression, SVM aims to find a hyperplane that best fits the data while minimizing the margin violations. SVM is particularly effective in capturing complex relationships and is robust against outliers. It uses a kernel trick to transform the feature space, allowing it to handle non-linear relationships.

**Convolutional Neural Network (CNN):**

Convolutional Neural Networks are deep learning models specifically designed for processing structured grid data, such as images or, in some cases, time-series data. CNNs use convolutional layers to automatically learn hierarchical features from the input data. In the context of bike usage prediction, a 1D CNN can capture temporal patterns and relationships in the time series data, making it well-suited for sequential prediction tasks.

In order to compare the models, I predicted the daily data from 2020-03-17 and compared it with the actual data till 2023-11-30 and got the following results for rmse and mae:

| | RMSE | MAE |
|---|---|---|
| Linear Regression | 5.838194234536177e-13 | 4.526561302055158e-13 |
| Ridge Regression | 4.0663149200424093e-10 | 3.4676939343964684e-10 |
| K-Nearest Neighbour | 23.822832233552678 | 14.227149659793664 |
| Long Short-Term Memory | 507.6216205378193 | 405.22885766788596 |
| Support Vector Machine | 613.268495963982 | 554.2841482806782 |
| Convolutional Neural Network | 814.5341578858752 | 613.1867349287772 |

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are metrics commonly used to evaluate the accuracy of a predictive model. They provide insights into how well the model's predictions align with the actual values in the dataset.

- RMSE measures the average magnitude of the prediction errors. It penalizes large errors more heavily than smaller errors.
- Lower RMSE values indicate better model performance, with 0 being a perfect fit (all predictions exactly match the actual values).
- RMSE provides a comprehensive understanding of how well a model predicts across the entire dataset.
- It is sensitive to outliers, making it useful when large errors need to be minimized
- MAE measures the average absolute difference between actual and predicted values.
- Like RMSE, lower MAE values indicate better model performance, with 0 being a perfect fit.
- MAE is less sensitive to outliers compared to RMSE, making it suitable when the impact of large errors needs to be minimized but not heavily penalized.

Both RMSE and MAE provide easily interpretable measures of prediction accuracy in the same units as the target variable. These metrics allow for easy comparison of different models or variations of a model. Lower values indicate better accuracy. During model development, RMSE and MAE can be used to compare the performance of various models and guide the selection of the most accurate one for a specific task.

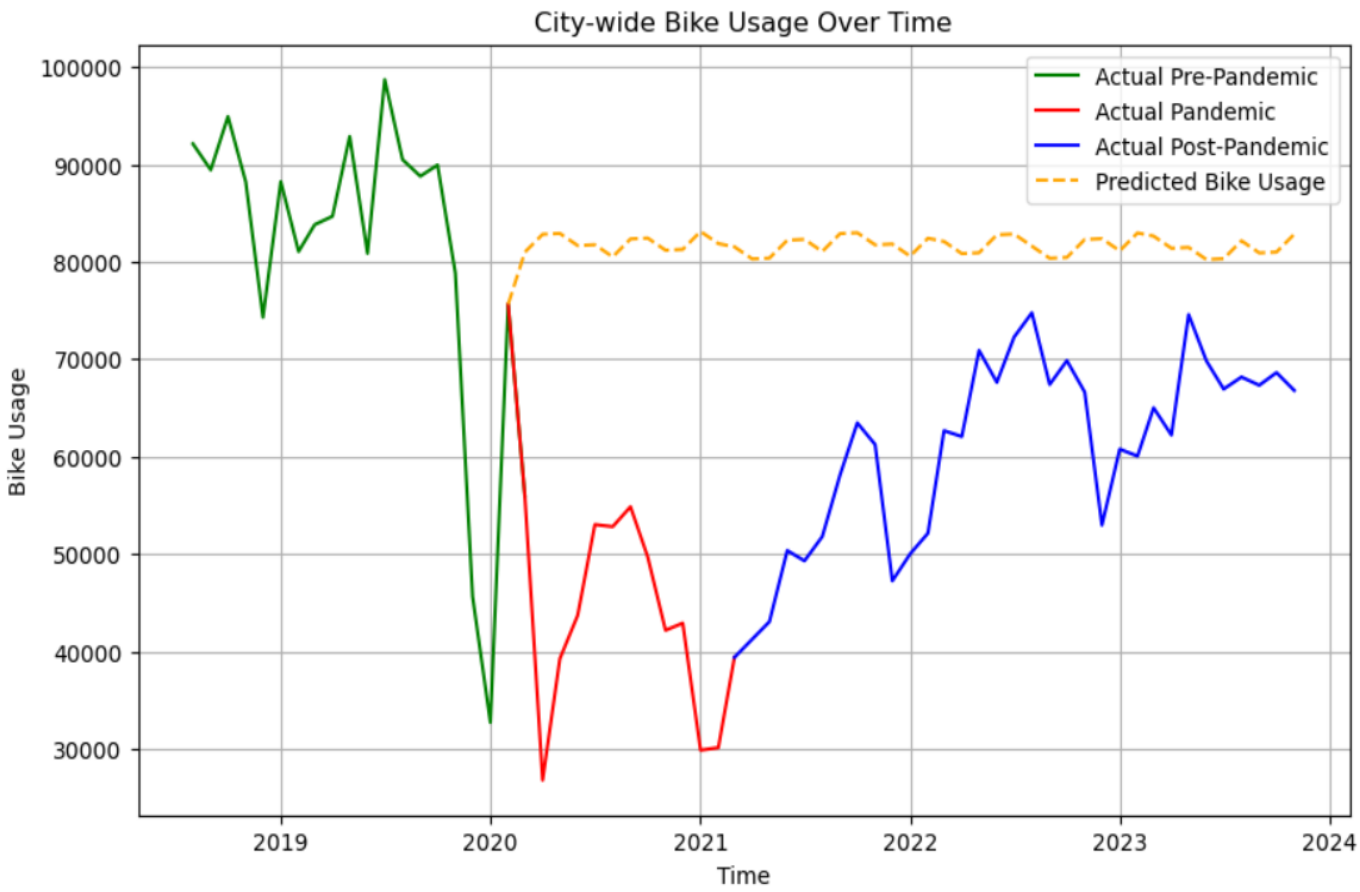
As seen from the table linear and ridge regression models demonstrate outstanding performance with extremely low prediction errors. KNN shows higher errors, suggesting challenges in capturing patterns. SVM and CNN models exhibit larger errors compared to the regression models, indicating potential difficulties in learning from the data or overcomplexity in the case of CNN.

When looking at the actual data plot in Graph 1 I observed that there is a sudden and odd drop in bike usage in June 2019, on close inspection of the data I found that the data after 2019-06-12 is missing and is not available in the original Dublin-bikes csv for June Q2. So, I first made daily bike usage prediction till June 30$^{th}$ to fill the gap.

I used Linear Regression as my preferred model to complete our tasks which are as follow:

**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.**

**Graph 2**




**Impact of the pandemic on city-bike usage for the pandemic period:**

The graph shows that city-bike usage in Dublin, Ireland, decreased significantly during the pandemic period. The average city-bike usage for the pandemic period was around 50,000 per day, compared to an average of 80,000 per day in the pre-pandemic period. This decrease is likely due to a number of factors, including:

- Restrictions on movement: During the pandemic, many people were restricted from moving around freely, either due to government lockdowns or personal concerns about the virus. This reduced the number of people who were able to use city bikes for commuting, leisure, and other activities.
- Reduced demand for travel: The pandemic also led to a decrease in overall travel demand, as many people were working from home and avoiding unnecessary trips. This also contributed to the decrease in city-bike usage.

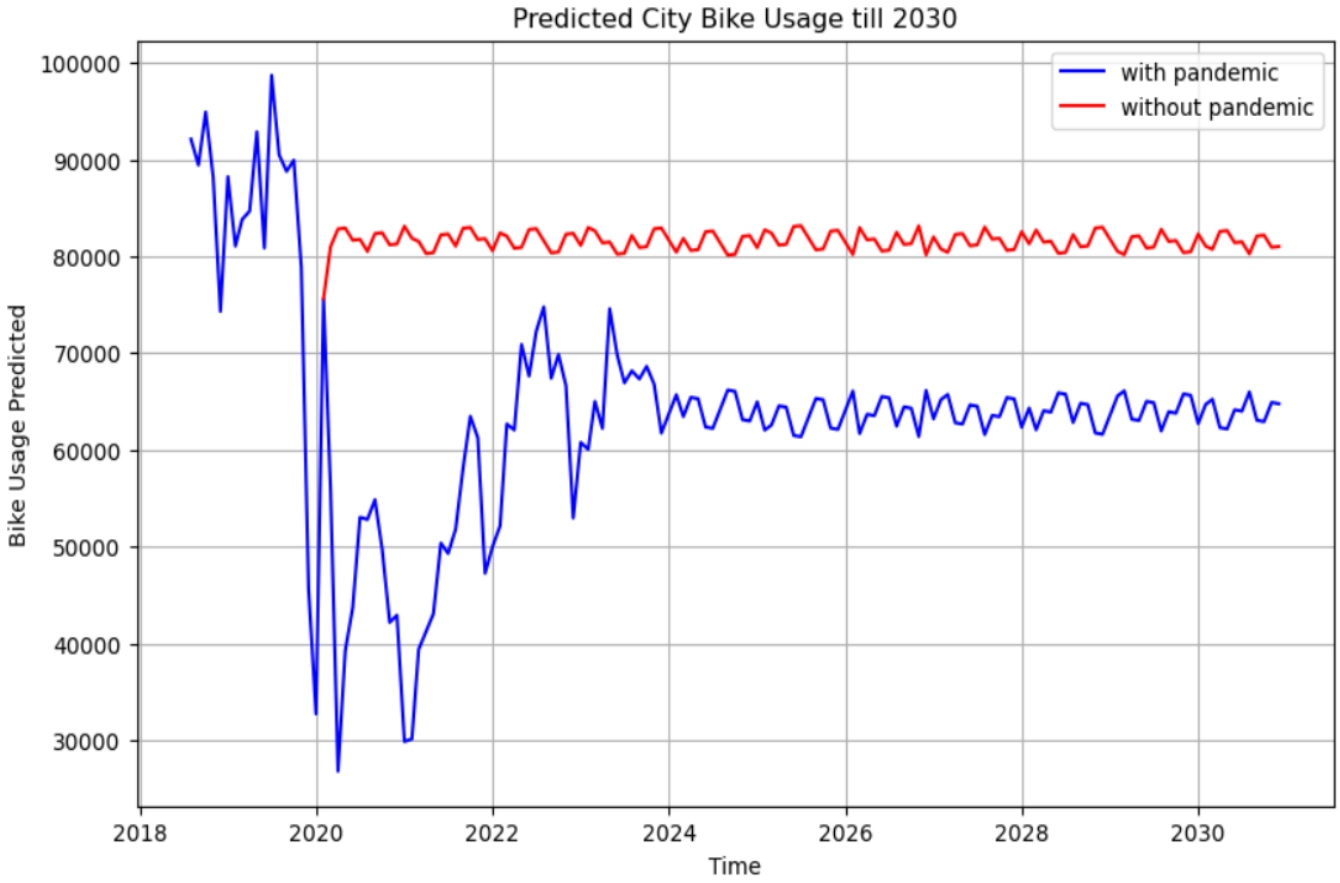
**Impact of the pandemic on city-bike usage for the post-pandemic period**

City-bike usage in Dublin has since recovered from the pandemic lows, but it has not yet returned to pre-pandemic levels. The average city-bike usage for the post-pandemic period has been around 60,000 per day. This is likely due to a number of factors, including:

- Continuing caution: Some people may still be cautious about using city bikes due to the pandemic. They may be concerned about touching shared surfaces or being in close contact with other people.
- Shift in travel patterns: The pandemic has led to some lasting shifts in travel patterns, as more people are now working from home and commuting less often. This may have reduced the demand for city bikes for commuting purposes.

**Predicting till 2030:**

**Graph 3**




The graph shows that city-bike usage in Dublin, is expected to continue to grow in the coming years, even with the impact of the pandemic. The blue line, which shows the predicted usage with the pandemic, shows that city-bike usage is expected to reach around 70,000 trips per day by 2030. This is slightly lower than the predicted usage without the pandemic (red line), which showed that city-bike usage could have reached around 80,000 trips per day by 2030.

The graph also shows that the pandemic has had a lasting impact on city-bike usage. The predicted usage with the pandemic is consistently lower than the predicted usage without the pandemic for the entire forecast period. This suggests that the pandemic has led to some structural changes in travel patterns, which are likely to persist even after the pandemic is over.

Despite the impact of the pandemic, the overall trend is positive for city-bike usage in Dublin. The predicted usage with the pandemic is expected to reach 70,000 trips per day by 2030 which is a significant increase from the current usage of around 60,000 trips per day. This shows that the city-bike network is expected to continue to grow, and more and more people are expected to use city bikes for transportation.

This is good news for the environment and for public health.

# Part - II

**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?**

A ROC curve, also known as the receiver operating characteristic curve, is a graphical tool used to evaluate the performance of a binary classifier. It plots the true positive rate (TPR) against the false positive rate (FPR) at different thresholds. The TPR is the proportion of positive samples that are correctly classified as positive, while the FPR is the proportion of negative samples that are incorrectly classified as positive.

Evaluating Classifier Performance

A ROC curve can be used to evaluate the performance of a classifier by comparing its curve to the curve of a baseline classifier. A baseline classifier is a simple classifier that does not use any information about the data to make predictions. A common baseline classifier is the "always predict negative" classifier, which always predicts that a sample is negative. This classifier has a TPR of 0 and an FPR of 1.

A classifier with a ROC curve that lies above the baseline classifier's curve is considered to be a better classifier. This is because the classifier is able to correctly classify more positive samples while also incorrectly classifying fewer negative samples.

Using ROC Curves Instead of Accuracy

In some cases, using an ROC curve instead of a classification accuracy metric can be more informative. This is because accuracy can be misleading when the classes in the data are not evenly distributed. For example, if a classifier is only used to identify a rare disease, then a high accuracy may not be meaningful if the classifier is also making many false positives.

ROC curves are also useful for comparing the performance of multiple classifiers. This is because they can be used to compare the classifiers at different levels of sensitivity (TPR) and specificity (1-FPR). For example, a classifier with a high TPR may be preferred if it is important to identify as many positive cases as possible, even if this means that more negative cases are also identified.

Interpreting ROC Curves

A ROC curve can be interpreted by looking at its AUC (area under the curve). The AUC is a measure of the overall performance of the classifier. An AUC of 1 represents a perfect classifier, while an AUC of 0.5 represents a random classifier.

The ROC curve can also be used to calculate the optimal threshold for the classifier. The optimal threshold is the point on the ROC curve where the TPR and FPR are equal. This threshold is the point that balances the trade-off between sensitivity and specificity.

**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.**

Example 1: Predicting house prices based on square footage

Linear regression assumes a linear relationship between the independent variable (square footage) and the dependent variable (house price). However, in the real world, the relationship between house price and square footage is not always linear. For example, houses with a lot of square footage may be less expensive per square foot if they are in a less desirable location.

Example 2: Predicting the number of sales of a product based on its price

Linear regression assumes that the relationship between the independent variable (price) and the dependent variable (number of sales) is a straight line. However, in the real world, the relationship between price and sales may be nonlinear. For example, the number of sales may increase slowly at first, but then increase more rapidly as price decreases. This could be because customers are more price sensitive at higher price points.

**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.**

In SVM, kernel refers to a function that transforms the input space into a new, higher-dimensional space where the data is linearly separable. This transformation allows SVM to classify data points that are not linearly separable in the original space. There are many different types of kernels available, including linear, polynomial, radial basis function (RBF), and sigmoid kernels. The choice of kernel depends on the specific characteristics of the data and the desired classification accuracy.

In CNNs, kernel refers to a small filter that is convolved with input data to extract features. The filter is typically a small matrix of weights that is learned during the training process. The convolution operation slides the filter across the input data, producing a feature map that contains information about the presence or absence of patterns in the data. CNNs use a variety of different types of kernels, including convolutional, pooling, and activation kernels.

why the use of kernels is useful in SVM and CNN models:

- SVM kernels allow SVM to classify data points that are not linearly separable. This is important for many real-world applications, where data often exhibits complex relationships.
- CNN kernels allow CNNs to extract features from input data that are relevant to the task at hand. This is important for tasks such as image classification, object detection, and natural language processing.

when the use of SVM kernels and CNN kernels is useful:

- SVM kernels are useful for tasks such as text classification, where the data is not linearly separable.
- CNN kernels are useful for tasks such as image classification, where the data contains spatial relationships.

different types of kernels used in SVM and CNN models:

- Linear kernel: The simplest type of kernel, which simply maps the input data to the same space.
- Polynomial kernel: A kernel that raises the dot product of the input vectors to a power.
- RBF kernel: A kernel that uses a Gaussian function to measure the similarity between input vectors.
- Sigmoid kernel: A kernel that uses a sigmoid function to measure the similarity between input vectors.
- Convolutional kernel: A kernel that is used to extract features from images.
- Pooling kernel: A kernel that is used to downsample the input data.
- Activation kernel: A kernel that is used to introduce non-linearity into the network.

**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.**

K-fold cross-validation, also known as leave-p-out cross-validation, is a resampling method used to evaluate the performance of a machine learning model on a dataset. It involves dividing the dataset into k folds, training the model on k-1 folds, and evaluating its performance on the remaining fold. This process is repeated k times, each time using a different fold as the test set.

The idea behind resampling is to simulate the process of using the model to make predictions on new, unseen data. By splitting the dataset into multiple folds, we can ensure that the model is trained on a representative sample of the data and that its performance is evaluated on a variety of data. This helps to reduce the risk of overfitting, which occurs when a model becomes too specific to the training data and does not generalize well to new data.

In k-fold cross-validation, the performance of the model is measured using the average of the performance scores from all k folds. This gives us a more robust estimate of the model's generalization performance than simply evaluating it on the entire training set.

Example:

Suppose we have a dataset of 100 examples and we want to evaluate the performance of a linear regression model on this data. Using k-fold cross-validation with k=5, we would divide the dataset into 5 folds of 20 examples each.

For each fold, we would train the linear regression model on the remaining 4 folds of data and evaluate its performance on the 5th fold. We would repeat this process 5 times, each time using a different fold as the test set.

The average of the performance scores from all 5 folds would then be used as an estimate of the model's generalization performance.

When to Use K-fold Cross-validation:

K-fold cross-validation is a powerful and versatile method for evaluating the performance of machine learning models. It is particularly useful when the dataset is relatively small or when the model is computationally expensive to train.

Here are some specific examples of when k-fold cross-validation is appropriate:

- When the dataset is small and cannot be effectively divided into a training and test set without introducing significant bias.
- When the model is computationally expensive to train and evaluating it repeatedly on the entire training set is impractical.
- When the model is complex and prone to overfitting.

When Not to Use K-fold Cross-validation:

K-fold cross-validation is not appropriate for all machine learning tasks. Here are some situations where it should be avoided:

- When the dataset is very large and dividing it into k folds would be impractical or time-consuming.
- When the model is sensitive to the order in which the data is presented.
- When the model is designed to handle time-series data.

In these cases, other methods for evaluating model performance, such as holdout validation or bootstrapping, may be more appropriate.