Here is a comprehensive set of revision notes from the class, focusing on the concepts of k-Nearest Neighbors (kNN):

Revision Notes: k-Nearest Neighbors (kNN)

Agenda

- kNN Sklearn Code
- Bias-Variance of kNN
- Hyperparameter Tuning
- Time Complexity of kNN
- kNN for Categorical Features
- Distance Metrics
- kNN for Real World
- kNN Imputation

kNN Fundamentals

- **kNN (k-Nearest Neighbors)** is a non-parametric, lazy learning algorithm used for both classification and regression.
- It works by finding the 'k' training samples closest in the feature space to the given test sample and making predictions based on the labels of its k-neighbors.

Working and Formula

1. **Calculate Euclidean Distance**: For every data point in the dataset, calculate the distance to the query instance. The Euclidean distance formula for two points $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ is given by: $distance = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ This formula can be adapted to Manhattan distance $(\sum |x_i - y_i|)$, depending on the context and data type【4:16†source】【4:6†source】.
2. **Sorting and Selecting Neighbors**: Sort all the calculated distances, and select the k data points.

as the prediction.

Choosing the Right k

- **Bias-Variance Trade-off**:
  - A smaller 'k' makes the algorithm sensitive to noise, leading to overfitting.
  - A larger 'k' can smooth out noise but may cause underfitting by over-generalizing the data【4:15†source】.

Handling Categorical Data with kNN

- Categorical data can be transformed using techniques like:
  - **One-Hot Encoding**: Essential for nominal data where order doesn't matter.
  - **Label Encoding**: Can be used when categorical data has an ordinal nature【4:10†source】.

Distance Metrics Considerations

- **Euclidean Distance**: Useful for less than 7 dimensions.
- **Manhattan Distance**: Preferable for dimensions between 7 and 15【4:16†source】.
- **Cosine Similarity**: Suitable for high-dimensional data beyond 15 dimensions【4:10†source】.

Real World Applications

- Generally used in interview scenarios rather than real world projects due to high computational complexity with large datasets.
- It can be used for basic comparison tasks when the dataset is small, or for educational purposes【4:7†source】.

Hyperparameter Tuning

- **Finding Optimal k**: Test the algorithm with different k values, choosing the one that offers the best performance across a validation set.

Locality Sensitive Hashing (LSH)

- LSH is used to speed up kNN by partitioning the data space into 'buckets' or grids, each storing the nearest neighbors. During testing, the feature space is consulted to rapidly provide predictions【4:1†source】【4:4†source】.

kNN Imputation

- Impute missing values in a dataset using kNN by finding k similar cases and averaging their values for numerical data, or using their mode for categorical data【4:19†source】.

Conclusion

kNN is a versatile algorithm that's relatively simple to implement but can be computationally expensive as the dataset size grows. It requires careful preprocessing, particularly with regard to distance measurement and categorical data handling. The understanding of kNN extends into recognizing the bias-variance trade-off and the importance of selecting a proper value for k, which greatly influences model accuracy.

These notes synthesize the key topics discussed in the class and will serve as an effective guide for understanding and implementing kNN in various scenarios.