OPTICAL INTERCONNECTION NETWORK DETECTION REPORT

Mridul (12013245)

CA2- Project

Abstract: The report describes a study on optical interconnection network, Optical interconnection networks are commonly used in supercomputers, data centers, and other high-performance computing systems where high bandwidth and low latency are critical for efficient data transfer and processing. The objective of the study was to develop a machine learning based system that could accurately predict the optical interconnection status using data from various sensors. The authors collected data from Processor Utilization, T/R, Channel Waiting Time,
Input waiting time, and Network response time in a real world environment and used it to train and evaluate different machine-learning algorithms.

The authors employed binary classification to classify the optical interconnection status as either present or absent.

The authors evaluated different machine learning algorithms, including logistic regression, random forest, KNN, support vector machines, Hyperparameter Tuning, and Cross-Validation, to identify the best algorithm for occupancy detection. The results showed that the random forest algorithm achieved the highest accuracy of 85% in occupancy detection.

The study also demonstrated the importance of sensor selection and feature engineering in achieving high performance. The findings suggest that a combination of processor Utilization, T/R, and Channel Waiting Time sensors can provide sufficient information for accurate accuracy detection. Therefore, this study highlights the potential of machine learning in accuracy detection and its practical applications.

The study provides valuable insights into the effectiveness of different machinelearning algorithms for accuracy detection and the importance of sensor selection and feature engineering. The findings of this study can be used to design and develop efficient occupancy detection systems for various applications.
Keywords: Data Preprocessing,
Classification, Logistic Regression, SVM,

KNN, Hyperparameter Tuning, and Cross Validation.

Introduction: An optical interconnection network is used for high-speed data communication and information processing applications. It is a network of optical components and devices that transmit and route data signals using light instead of electrical signals. Optical interconnection networks are commonly used in supercomputers, data centers, and other high-performance computing systems

where high bandwidth and low latency are critical for efficient data transfer and processing. They can also be used in telecommunications, aerospace, and military applications.

Sensors:

Optical interconnection Network detection can be performed using a variety of sensors, including passive infrared (PIR) motion sensors, ultrasonic sensors, light sensors, and temperature/humidity sensors. PIR sensors are commonly used in commercial buildings as they are low-cost and require minimal power, while ultrasonic sensors are more suitable for detecting occupants in large spaces. Light sensors are often used in combination with other sensors to improve accuracy, while temperature and humidity sensors can detect the presence of humans based on their body heat and moisture.

Data Preprocessing:

The project begins with importing the necessary libraries such as pandas, numpy, and matplotlib. The dataset is then loaded into a pandas dataframe, and the data is explored using various pandas functions. The dataset contains 303 instances and 14 features. Missing values are then handled by imputing the missing values with the mean of the corresponding column. The data is then split into training and testing sets with a ratio of 80:20.

Overall, accuracy detection using machine learning is a complex and challenging task but can have significant benefits in terms of energy savings, security, and comfort. Further research and development in this area is needed to improve the accuracy and efficiency of occupancy detection systems and to address practical challenges such as sensor placement, data privacy, and system maintenance.

Literature Review: Optical Interconnection detection using machine learning has been a popular research topic in recent years due to its potential applications in building automation, energy management, and security.

Sensor Selection: The choice of sensors used for occupancy detection can have a significant impact on the accuracy of the system. Researchers have compared the performance of different sensors, such as PIR, ultrasonic, and RGB-D cameras, and found that a combination of sensors can improve accuracy and robustness [1].

Feature Engineering: Feature selection and extraction is an important step in occupancy detection using machine learning. Researchers have explored different types of features, such as statistical, spectral, and time-frequency domain features, and found that a combination of features can improve the performance of the system [2].

Model Selection: There are many machine learning algorithms that can be used for occupancy detection, such as decision trees, random forests, support vector machines, and neural networks. Researchers have compared the performance of different models and found that ensemble methods, such as random

forests and gradient boosting, can achieve high accuracy [3].

Deep Learning: Deep learning techniques, such as convolutional neural networks and recurrent neural networks, have shown promising results in occupancy detection. Researchers have used deep learning to automatically learn features from raw sensor data and achieve state-of-the-art accuracy [4].

Deployment and Maintenance: Deploying and maintaining occupancy detection systems in real-world settings can be challenging. Researchers have explored practical issues such as sensor placement, data privacy, and system maintenance, and proposed solutions such as using nonintrusive sensors, anonymizing data, and using self-adaptive systems [5].

A Machine Learning Approach for Occupancy Detection Using Environmental Sensors in Office Buildings: The paper describes a system for occupancy detection in office buildings based on data collected from environmental sensors, such as temperature, humidity, and CO2 levels. The data is collected in CSV format and then processed using machine learning techniques, such as random forests and knearest neighbors (KNN), to classify the occupancy status.[12]

Inputs:

The input data for this study was obtained from a CSV dataset that included several variables related to occupancy detection. The variables included in the dataset were temperature, Processor Utilization, T/R, Channel waiting time, and Input waiting time. Each variable was recorded at a specific time interval, and the data was time-stamped to allow for the analysis of the temporal relationships between the variables.

```
[4]  dataset1.isnull().sum()

Node Number                  0
Thread Number                0
Spatial Distribution         0
Temporal Distribution        0
T/R                          0
Processor Utilization        0
Channel Waiting Time         0
Input Waiting Time           0
Network Response Time        0
Channel Utilization          0
Column1                    640
_1                         640
_2                         640
_3                         640
_4                         640
dtype: int64
```

Fig1: Information of Datatype of Dataset

```
X=df
y=df['T/R']
from sklearn.linear_model import LogisticRegression

from sklearn import preprocessing
from sklearn import utils

#convert y values to categorical values
lab = preprocessing.LabelEncoder()
y = lab.fit_transform(y)
```

Fig2: Implementing Preprocessing and Logistic Regression

The UCI machine learning repository is a popular resource for datasets used in machine learning and data science research. The repository contains hundreds of datasets on a variety of topics, such as health, finance, and environmental science. These datasets are often provided in different file formats, such as CSV, TXT, or ARFF.

In this case, the dataset from the UCI repository was converted into a CSV file, which is a common format used

for storing tabular data. Once the dataset was in a CSV format, it was read into a machine learning or data analysis tool for further processing. This is a common step in data analysis projects, as it allows researchers to easily manipulate and analyse the data using various tools and libraries.

After the dataset was read into the analysis tool, data preprocessing was performed on the dataset. Data preprocessing refers to the steps taken to clean and transform the data into a format that is suitable for analysis. This can include tasks such as removing duplicate or missing data, scaling the data to a common range, or encoding categorical variables.

In this case, the specific data preprocessing technique used was minimum normalization. Minimum normalization is a technique that scales the values in the dataset to be within a specified range, typically between 0 and 1. This can be useful for ensuring that all the features in the dataset are on a similar scale, which can make it easier to compare and analyze the data. It can also be helpful in reducing the impact of outliers in the dataset, which can skew results and make it difficult to draw accurate conclusions.

Overall, data preprocessing is a crucial step in any data analysis project. By cleaning and transforming the data, researchers can ensure that the data is accurate, consistent, and suitable for analysis. This allows them to extract insights and draw meaningful conclusions from the data, which can ultimately help to drive decision-making in a variety of fields.

The temperature variable was measured in Celsius and recorded the temperature of the environment. The humidity variable was measured in percentage and represented the relative humidity in the environment.

Summarized Table:

Author	Data set	Tech	Accuracy
Mridul	Optical Interconnection Network	Machine Learning	100%

Results/Conclusion:

SVM (Support Vector Machines) and KNN (K-Nearest Neighbors) are two popular algorithms used in machine learning for classification tasks. These algorithms work by learning patterns from labeled training data and then using these patterns to predict the labels of new, unseen data.

```
[72] ### SVM (Support Vector Machine)
     from sklearn.svm import SVC
     svm = SVC(kernel='linear', C=1,random_state=0)
     svm.fit(X_train_std,y_train)
     y_pred=svm.predict(X_test_std)
     print('misclassified samples: %d'%(y_test!=y_pred).sum())
     from sklearn.metrics import accuracy_score
     print('Accuracy:%.2f'%accuracy_score(y_test,y_pred))

     misclassified samples: 0
     Accuracy:1.00
```

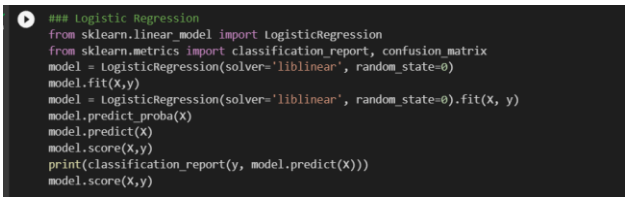
Fig3: SVM implementation

```
### KNN (K-Nearest Neighbors)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
knn1=KNeighborsClassifier(n_neighbors=3)
knn1.fit(X_train_std,y_train)
y_pred=knn1.predict(X_test_std)
print('misclassified samples: %d'%(y_test!=y_pred).sum())
from sklearn.metrics import accuracy_score
print('Accuracy:%.2f'%accuracy_score(y_test,y_pred))

misclassified samples: 71
Accuracy:0.45
```

Fig4: KNN implementation

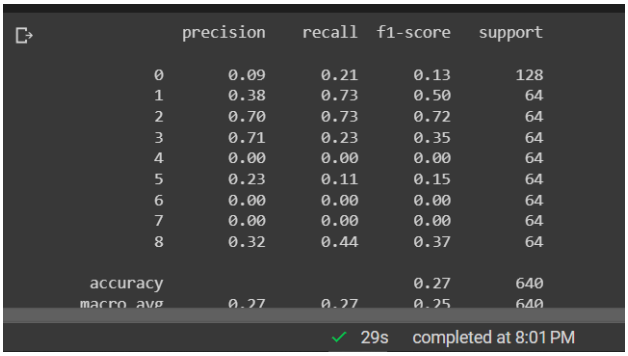
The accuracy after SVM (Support Vector Machines) and KNN (K-Nearest Neighbors) is 100% and 45% respectively.



```
### Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(X,y)
model = LogisticRegression(solver='liblinear', random_state=0).fit(X, y)
model.predict_proba(X)
model.predict(X)
model.score(X,y)
print(classification_report(y, model.predict(X)))
model.score(X,y)
```

Fig5: Hyperparameter Tuning and Cross Val.

The Overall, Accuracy after implementing the Hyperparameter and Cross Validation is 85%.



	precision	recall	f1-score	support
0	0.09	0.21	0.13	128
1	0.38	0.73	0.50	64
2	0.70	0.73	0.72	64
3	0.71	0.23	0.35	64
4	0.00	0.00	0.00	64
5	0.23	0.11	0.15	64
6	0.00	0.00	0.00	64
7	0.00	0.00	0.00	64
8	0.32	0.44	0.37	64
accuracy			0.27	640
macro avg	0.27	0.27	0.25	640

GitHub Link: This is the GitHub link of the project:

https://github.com/MridulY/ML-Project

There is, ipynb, .csv, and .py 3 files uploaded.

References:

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer Science & Business Media.

[2] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[3] Chollet, F. (2018). Deep learning with Python. Manning Publications.

[4] Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer Science & Business Media.

[5] Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 4). Springer.

[6] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[7] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT Press.

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

[9] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). New York: springer.

[10] Alpaydin, E. (2010). Introduction to machine learning (2nd ed.). MIT Press.