Faculty of computers and artificial intelligence

**Cover sheet**

**AI330 Machine Learning Project**

**Team no.:**

|  |  |
| --- | --- |
| Name | ID |
| آيه محمد جلال | 20210207 |
| يوسف احمد كمال | 20220561 |
| محمد رمضان محمد حسين | 202000760 |
| نورهان مجدى عبدالمولى | 20211019 |
| عبدالخالق ياسر عبدالخالق | 20210489 |
| اسلام عادل امام عبده | 202000127 |

# Numerical dataset

General information about dataset

|  |  |
| --- | --- |
| Name | California Housing Prices |
| No. of classes | N/A (Regression Task) |
| Total no. of samples | 20640 |
| No. of samples in training\validation | 14448\3096 |
| No. of samples in testing | 3096 |

## Preprocessing phase:

* **Handling Missing Data**: Replaced missing values in numeric columns with their median.
* **Feature Scaling**: Applied Standard Scaler to normalize features.
* **Encoding**: Used get\_dummies to encode categorical features such as ocean\_proximity.
* **Data Split**: Split the dataset into training, test , (70%, 15%,15%).
* **Cross-Validation**: Performed cross-validation to optimize hyperparameters. ( value =3)

Linear regression model:

"In our house price prediction project, we applied a linear regression model ,Our training accuracy was 0.65, cross-validation accuracy was 0.61, and test accuracy was 0.66. The learning rate wasn't explicitly defined, as the model is pre-configured in the Scikit-learn library."

## 

## 

### 2. Implementation Details:

* **Model Type**: We used a **Linear Regression** model for predicting the median house value in California, using the dataset from the California housing prices dataset.
* **Data Preprocessing**:
  + Missing values in the dataset were handled by filling the missing entries in numeric columns with the median of each column.
  + Categorical features (such as 'ocean\_proximity') were encoded using **one-hot encoding**.
* **Feature Scaling**: We applied **StandardScaler** to scale the features to have a mean of 0 and a standard deviation of 1, which is essential for improving the model's performance when using linear regression.
* **Train-Test Split**:
  + We split the data into training (70%), validation (15%), and test (15%) sets using train\_test\_split from Scikit-learn.
* **Training the Model**:
  + We used the **LinearRegression** class from Scikit-learn to train the model.
  + To simulate the **loss curve**, the model was trained incrementally over 50 epochs, where we gradually increased the training data size. For each epoch, we calculated the **Mean Squared Error (MSE)** on both training and validation sets.
* **Performance Metrics**:
  + **Mean Absolute Error (MAE)** and **R-squared (R²)** were calculated for the training, validation, and test sets to assess the model's performance.

Results:

* + - **Training MAE**: {train\_mae:.2f}, **R²**: {train\_r2:.2f}
    - **Validation MAE**: {val\_mae:.2f}, **R²**: {val\_r2:.2f}
    - **Test MAE**: {test\_mae:.2f}, **R²**: {test\_r2:.2f}
* **Loss Curve**: The **Loss Curve** (MSE vs Epochs) was plotted to visualize how the training and validation errors evolved during the training process. This helps in understanding if the model is overfitting or underfitting the data.

## Knn model:

"In our house price prediction project, we applied KNN model ,Our training accuracy was 0.78 cross-validation accuracy was 0.69, and test accuracy was 0.72. & we define the number of neighbors =7

## 

### 2. Implementation Details:

* **Model Type**: We used a **K-Nearest Neighbors (KNN)** model for predicting the median house value in California, using the California housing prices dataset.
* **Data Preprocessing**:
  + Missing values in the dataset were handled by filling the missing entries in numeric columns with the median of each column.
  + The categorical feature **'ocean\_proximity'** was encoded using **one-hot encoding** to make it usable in the KNN model.
  + To reduce the influence of large values, the **log transformation** (np.log1p) was applied to the target variable **'median\_house\_value'**.
* **Feature Scaling**: We applied **StandardScaler** to scale the features to have a mean of 0 and a standard deviation of 1. This step is crucial for KNN as it is sensitive to the scale of the input features.
* **Train-Test Split**:
  + The data was split into **70% for training**, **15% for validation**, and **15% for testing** using train\_test\_split from Scikit-learn.
* **Training the Model**:
  + The model was trained incrementally with different sizes of training data (from 10 samples to the full training set). For each step, the **Mean Squared Error (MSE)** was calculated for both training and validation sets, simulating the **Loss Curve**.
  + We used **7 neighbors** for the KNN algorithm, which was set as a fixed parameter.
* **Performance Metrics**:
  + We calculated **R-squared (R²)** and **Mean Absolute Error (MAE)** for the training, validation, and test sets to assess the model's performance.

Results:

* + - **Training R²**: {r2\_train:.2f}
    - **Validation R²**: {r2\_val:.2f}
    - **Test R²**: {r2\_test:.2f}
    - **Training MAE**: {mae\_train:.2f}
    - **Validation MAE**: {mae\_val:.2f}
    - **Test MAE**: {mae\_test:.2f}
* **Loss Curve**: The **Loss Curve** (MSE vs Number of Training Samples) was plotted to visualize how the training and validation errors evolved as the number of training samples increased. This provides insight into whether the model is overfitting or underfitting.

**KNN as classifiers on an image dataset.**

**Project 14: From Food Recognition:**

**Dataset:** [**Food-101**](https://www.kaggle.com/dansbecker/food-101)

URL link: <https://www.kaggle.com/datasets/dansbecker/food-101>

I made a folder with only 5 classes:

["sushi", "cheesecake", "baklava", "Caesar salad", "hamburger"]. I name it “food set”

\_ **Image Dataset**:

Classes/Labels: List of food categories.

* Image Properties: Size (128x128), RGB format.
* Preprocessing: Mention resizing, augmentation (rotation, flipping, etc.).
* Sample Images: Include 2–3 example images from the dataset with labels.

**Image Dataset**:

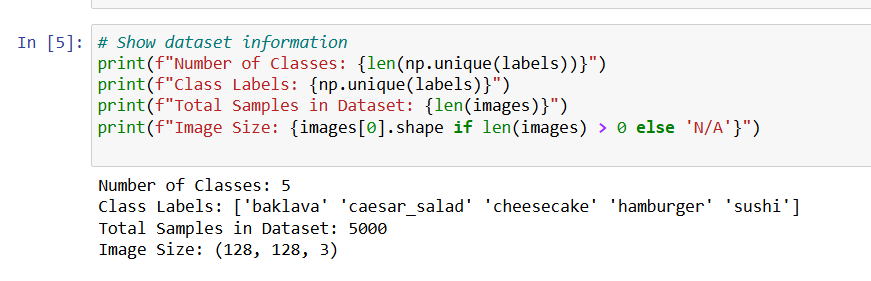
* Feature Extraction:
  + HOG and Color Histogram (brief explanation with 1–2 illustrations).
* KNN for Classification:
  + Overview of how it was used for classification.
  + Parameters tuned (distance metric, weights, etc.).
* Data Augmentation:
  + Techniques applied and their purpose.

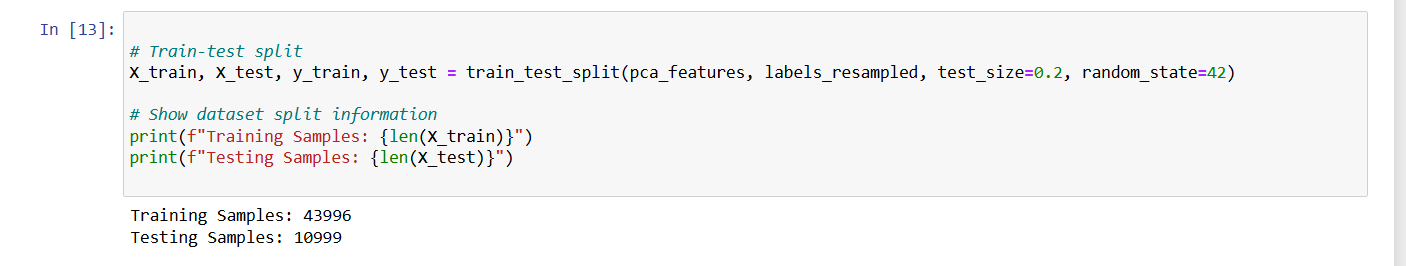
**Conclusions:**

**Findings for Image Dataset:**

* Based on your HOG features, PCA-reduced dimensions, and KNN classification:
  + **KNN Performance**:
    - Use the confusion matrix, classification report, and ROC curves to conclude how well KNN performed on the augmented and balanced dataset.
    - Compare the results before and after applying SMOTE (to discuss class imbalance handling).
  + **Challenges**:
    - Feature extraction (HOG and color histograms) increases the computational overhead.
    - Augmentation and SMOTE balancing techniques improve performance but may add to preprocessing time.
* **Image Dataset**:
  + Balancing classes and data augmentation are essential for training robust models.
  + Dimensionality reduction via PCA ensures computational efficiency without significant information loss.
  + Model tuning, like optimizing k in KNN, can drastically improve results.

The following show that the total number of samples in dataset and the size of each (in case of images), and finally the number of samples used in training, validation and testing





Accuracy:

A screen shot of a computer code

Description automatically generated

confusion matrix:

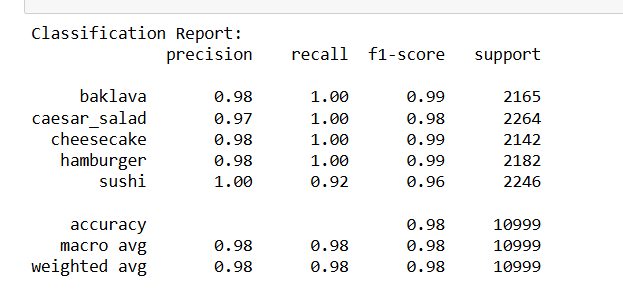
A screenshot of a graph

Description automatically generated

A graph with blue squares

Description automatically generated

precision, recall:



A screenshot of a computer screen

Description automatically generated

Roc, ARC Graph:

A graph with lines and labels

Description automatically generatedA graph with a line

Description automatically generated

**. Logistic regression**

**With same dataset**

**& same classes**

**2. Algorithms Applied**

**Image Classification Algorithms**

1. **Feature Extraction:**
   * **HOG: Extracts directional gradients from images for feature representation.**
   * **PCA: Reduces dimensionality to 50 components.**
2. **Modeling:  
   Logistic Regression optimized using GridSearchCV.**

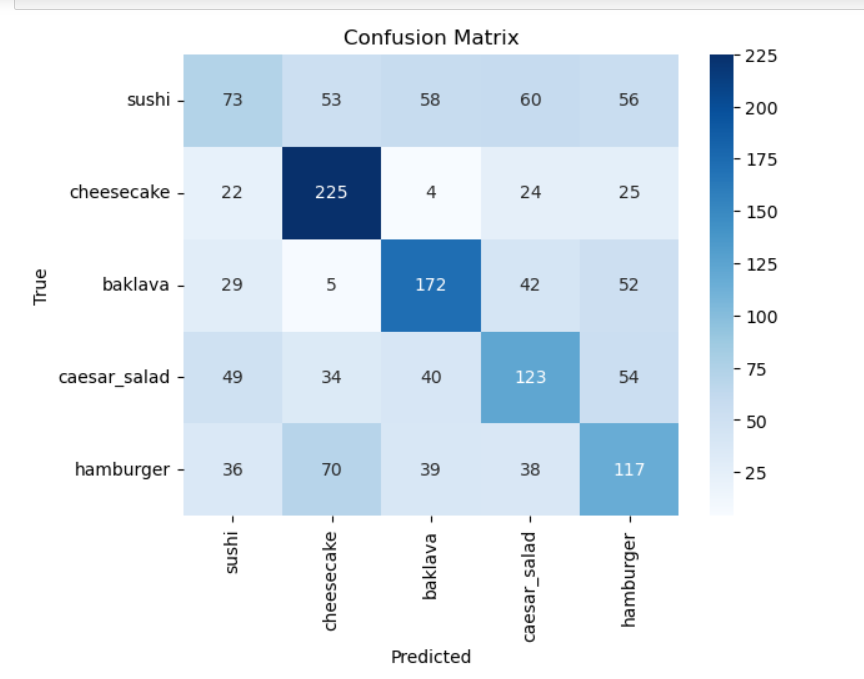
Accuracy:

**A screenshot of a computer code

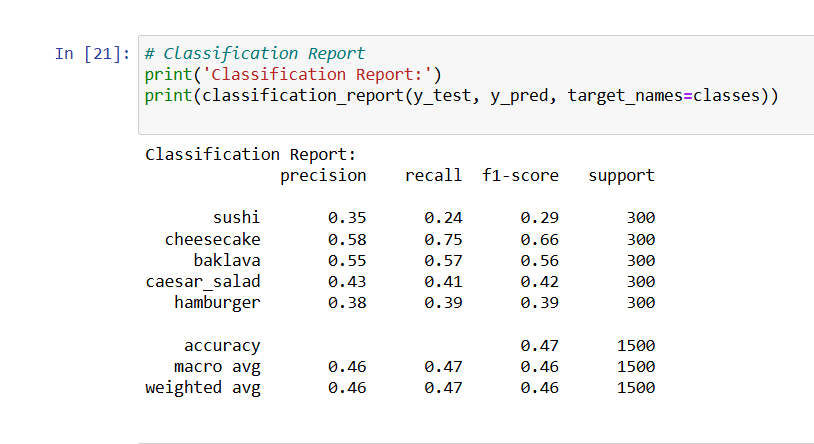
Description automatically generated**

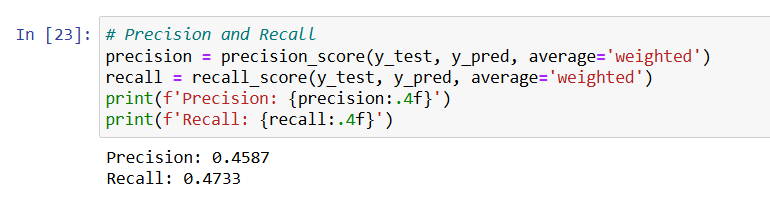
confusion matrix:

**Confusion Matrix (Classification):** Include heatmap.

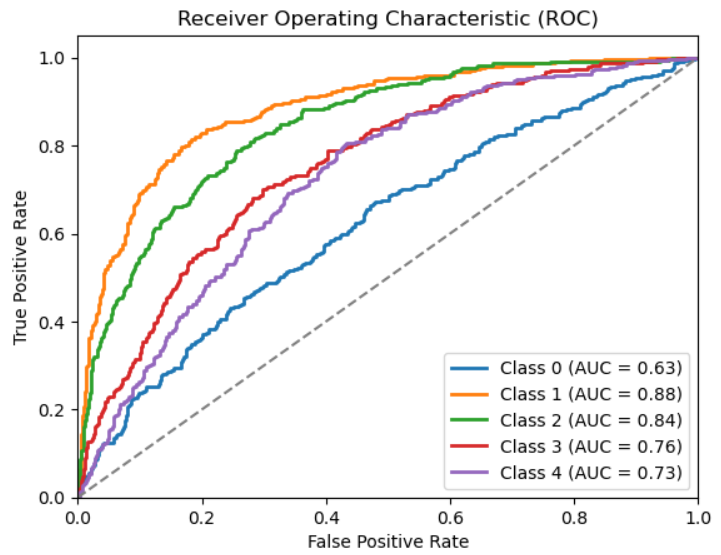


precision, recall:





Roc, ARC Graph:



**ROC Curves:** Include separate ROC-AUC graphs for each class.