

Overview

Numerical dataset

- Name: California Housing Prices
- Link: <https://www.kaggle.com/datasets/camnugent/california-housing-prices/data>

The dataset includes demographic and geographical data about various districts in California, such as population, median income, housing prices, and proximity to the ocean. While the data is not suited for predicting current housing prices, it serves as an excellent resource for understanding fundamental machine learning concepts and techniques. It has been featured in the book Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron, making it a popular choice for beginners in the field.

➤ Columns/Features:

- 1) longitude: measure of how far west a house is; a higher value is farther west
- 2) latitude: measure of how far north a house is; a higher value is farther north
- 3) housing_median_age: Median age of a house within a block; a lower number is a newer building
- 4) total_rooms: Total number of rooms within a block
- 5) total_bedrooms: Total number of bedrooms within a block
- 6) population: Total number of people residing within a block
- 7) households: Total number of households, a group of people residing within a home unit, for a block
- 8) median_income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
- 9) median_house_value[target]: Median house value for households within a block (measured in US Dollars)
- 10) ocean_proximity: Location of the house w.r.t ocean/sea

- Dataset originally isn't cleaned so it needs to be preprocessed:
 - 1) The total_bedrooms column has 207 missing values.
 - 2) The median_house_value column has 1071 rows with outlier values.

- Preprocessing :
 - 1) the total_bedroom got filled with the mean of the column.
 - 2) the outliers in the 1071 rows got dropped.

- Dataset size: 20640 rows and 10 columns.
- After preprocessing: 19569 rows and 10 columns.

Image dataset

The dataset used in this project is the Fruit 360 dataset. It contains over 90,000 labeled images of various fruits captured from different angles, classified into 141 classes. In this project, we used 5 classes from the dataset: apple, banana, eggplant, orange, and pepper. Collectively, they include over 3000 images, split between training and testing sets.

Algorithms

Linear Regression

- Supervised algorithm that models relationship between predictors(x) and target(y).
- Implementation process:
 - 1) Data preparation: splitting data into train and test sets:
training dataset = 80% = 15655 rows, testing dataset = 20% = 3914 rows
 - 2) Data training: using `linearregression()` and `fit(xtrain,ytrain)>>fitting` observed data
 - 3) Predicting:
using `predict()` : $\text{formula} = \text{ypredict} = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$
 w = coefficients learned during training
 - 4) Evaluation: from sklearn:

`importing from sklearn.metrics import r2_score`

`importing from sklearn.metrics import mean_squared_error as mse,
mean_absolute_error as mae`

1. Mean Squared Error (MSE): Measures average squared errors:

$$1/n * \sum(\text{ypred} - \text{ytrue})^2$$

2. Mean Absolute Error (MAE): Measures average absolute errors:

$$1/n * \sum(\text{ypred} - \text{ytrue})$$

3. R² Score: Evaluates the proportion of variance explained by the model (closer to 1 indicates better performance).

$$1 - \sum(\text{ypred} - \text{ytrue})^2 / \sum(\text{ytrue} - \text{ymean})^2$$

values: 1:perfect 0:predicts the mean only

<0: Model performs worse than predicting the mean

K-Nearest Neighbor (KNN) Algorithm

It makes predictions based on distances on nearest data points = (k) steps:

1) Data preparation: splitting data into train and test and validation sets:

training dataset = 80% = 15655 rows

test dataset = 10% = 1957 rows

validation dataset = 10% = 1957 rows

2) Sata training and validating:

2.1 at first i make an initial model with $k=2$ and make validation prediction based on this model $k=2$ and calculate validation mse and validation r^2 (to compare them with the result with the training predictions and choose the best model)

2.2 make a loop for k starting from $k=3$ to $k=30$ and make models for each k till i get the best (largest r^2 > near to 1) and best mse (smallest)

2.3 now we can make our best model with the the k we got

2.4 we choose the $p=1$ manhattean distance and do our fitting again with $k=13$ (best model)

3) Testing: we predict using `best_model.predict(x_test)`

4) Use evaluation metrics to calculate accuracy of model:

1. Mean Squared Error (MSE)

2. Mean Absolute Error (MAE)

3. R^2 Score

	Linear Regression Model	K-NN Model
MSE	3577840331.4375467	2701256623.3172107
MAE	44737.333750507874	36254.13352462561
R2 Score	0.6125661349874114	0.7065522159739022

Therefor: The KNN model is generally better.

Logistic Regress

In this project, we used multinomial logistic regression algorithm. It is a supervised machine learning algorithm that is used for classification where the purpose is to predict the probability that an instance belongs to a given class or not. The model transforms the linear regression function output into categorical value output by using the SoftMax function, mapping any real-valued number into the range (0, 1).

Number of training samples: 2333

Number of testing samples: 780

Each image is of size: (64x64x3)

K-Nearest Neighbor (KNN) Algorithm for classification

In this project, we applied the K-Nearest Neighbors (KNN) algorithm for multi-class classification to predict the type of fruit based on its visual features, specifically its color.

Key details of the implementation:

1. Data Preparation:

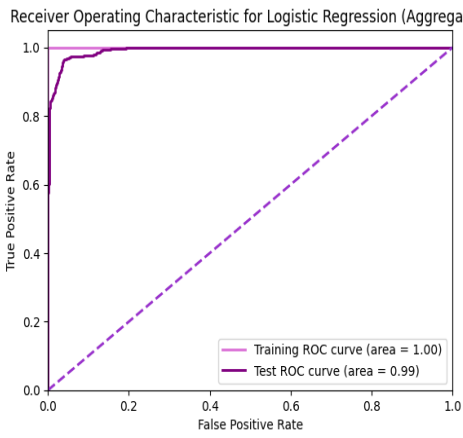
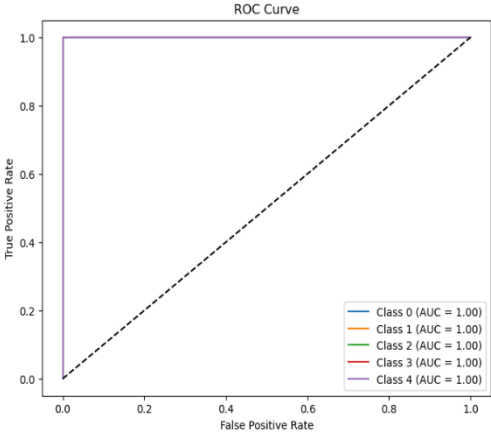
- The dataset consists of images of fruits, and the images for each fruit are stored in separate folders. Each image is resized to 64x64 pixels, and the pixel values are flattened into a 1D array to create feature vectors.
- Defined a dictionary by mapping each fruit class to a numeric label:
 - Apple: 0
 - Banana: 1
 - Eggplant: 2
 - Orange: 3
 - Pepper: 4

2. Training-Testing Split:

- The dataset was split into training (60%) and testing (40%) sets. 40% of the dataset was further divided into validation (20%) and test (20%) sets using the `train_test_split`
 - Training set: 1867 samples
 - Validation set: 623 samples
 - Test set: 623 samples
- KNN Classifier:
 - The K-Nearest Neighbors (KNN) algorithm was chosen with `n_neighbors=3`, using the Minkowski distance with parameter `p=2` (Euclidean distance).
 - The model was trained on the training data (`x_train`, `y_train`) using the `fit()` method of the `KNeighborsClassifier` class.

3. Conclusion:

Throughout this process, I took every measure to ensure the K-Nearest Neighbors (KNN) model performed optimally and was not subject to issues such as overfitting. By splitting the dataset into training, validation, and test sets, performing cross-validation to check if the accuracy will decrease or not but still the same without any changing, and evaluating multiple performance metrics such as accuracy, precision, recall, ROC, AUC, and confusion matrix, I ensured that the model was both robust and accurate. Additionally, the model's ability to generalize well to new data, as evidenced by the high performance on both the training and validation sets, suggests that the model is not overfitting and is ready for deployment.

	Logistic Regression Model	K-NN Model
Accuracy	90.5%	100%
Precision	92.3%	1.00
Recall	90.5%	1.00
Loss	0.568	2.2204460492503136e-16
Confusion Matrix	<pre>[[122 0 0 42 0] [0 152 0 0 0] [0 0 124 0 32] [0 0 0 160 0] [0 0 0 0 148]]</pre>	<pre>[[143 0 0 0 0] [0 126 0 0 0] [0 0 135 0 0] [0 0 0 103 0] [0 0 0 0 116]]</pre>
ROC Curve	<p>Receiver Operating Characteristic for Logistic Regression (Aggregated)</p> 	<p>ROC Curve</p> 
AUC	0.99	1.00