<u>Word Report: Programming Assignment 1</u>

<u>Subject: Machine Learning (CSE 6363 – 004)</u>

<u>UTA ID: 1002073011</u>

<u>Student Name: Prayag Axaykumar Parikh</u>

Zip Includes: Total 12 files:

1) Hayes_roth_scratch.ipynb [my implementation]
2) Hayes_roth_knn_scipy.ipynb [Scikit K-fold]
3) Car_evaluation_python_scratch.ipynb [my implementation]
4) Car_evaluation_scikit_knn.ipynb [Scikit K-fold]
5) Breast_cancer_scratch_python.ipynb [my implementation]
6) Breast_Cancer_scikit_kfold.ipynb [Scikit k-fold]
7) Hypothesis_testing.ipynb [Pair T-testing]
8) Final_Word_Report_PA1.docx
9) Final_Word_Report_PA1.pdf
10) 3 dataset files

IMPORTANT NOTES:

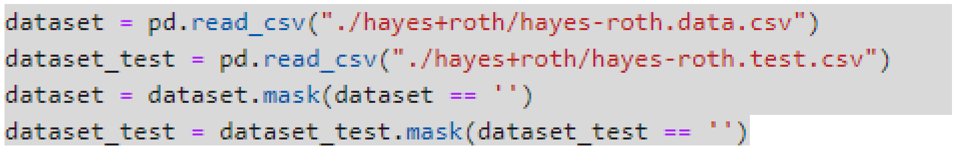
1. Everything in all .ipynb files is organized you just need to run the code in Jupyter Notebook.
2. From the EXTENSIONS implementation I've covered below things:
   a. **Tune KNN**: More values for k such as 3, 5, sqrt(dataset_size) and all explained above, check code outputs for results.
   b. **Data Preparation:** I've used Standarscaler method, data splitting using iloc method, and normalization when needed.
   c. **Different Datasets:** All 6 programs are working perfectly well for all given 3 datasets, both scratch and scikit KNN models.
3. For better reading experience please use the attached PDF version of same WORD file.
4. If you face any challenges in the code, then kindly contact me. Thank you!

Before I start, I wanted to describe that in k-NN, k is the number of 'nearest neighbors' selected to classify new data point. For all my implementations, I've taken two values of k. First is k=3 because it gives correct output with good accuracy most of the times, and another is k=sqrt(training_dataset_size), which is mostly preferable for large size datasets such as Car_Evaluation among given 3 datasets. I'll provide all the links to reference at the end.

Hayes Roth Dataset

1. **Hayes Roth Dataset My Implementation:**

- Dataset is very simple with total 6 columns, where 1st and 2nd columns, name and hobby respectively are not playing part in Class decision, therefore during calculations, I've considered C-E or 3-to-5 columns only.
- In KNN model predictions made based on the k nearest neighbors, therefore our whole concentration would be to find out k neighbors for the new Observation (data point).
- For all implementations, just for the dataset import, I've used Pandas library read_csv method.



```
dataset = pd.read_csv("./hayes+roth/hayes-roth.data.csv")
dataset_test = pd.read_csv("./hayes+roth/hayes-roth.test.csv")
dataset = dataset.mask(dataset == '')
dataset_test = dataset_test.mask(dataset_test == '')
```

- I've splitted the dataset into X_train, y_train(true_class), Y_train, y_test datasets, where number of rows are 132, 132, 28, 28 respectively.

From this point I'll describe the model KNN methods:

- Step1: To find the k-nearest neighbors, first we need to define what is 'near' and how to measure it. Here, I'm using Euclidean Distance as the distance measure to determine 'near', which is: the distance between two vectors such as new_observation_row = [u1,u2,u3,u4…] and another vector v=[v1,v2,v3,v4,….], and distance between them is:

  Euclidean_distance = sqrt((u1-v1)^2 + (u2-v2)^2 + (u3-v3)^2….)

  *This formula is used to find distance between new _observation and all other rows in the dataset
- Step 2: After finding distance between new_observation and all other rows we sort the distance array and fetch the 'top k' elements(neighbors)
- Step 3: In this last step, since we have the top-k nearest rows in an array, these predict_classification method will count the majority votes for all the unique classes. If class 2 has maximum number of votes in the neighbors list then the new_observation class is predicted as class 2.

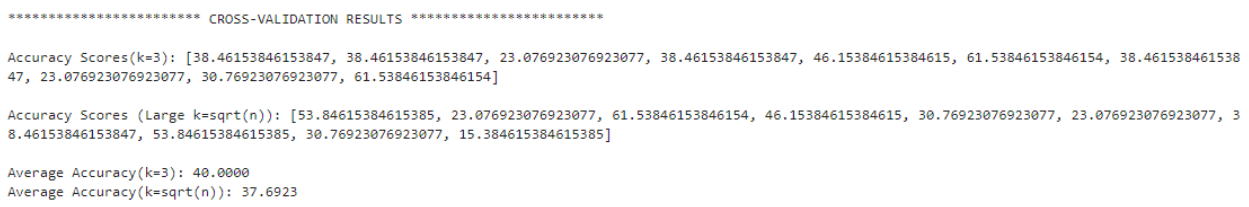
Remember, this is predicted class and actual outputs may vary and therefore we use 10 Fold accuracy model to find out performance between Actual output vs Predicted class output. Now, 10-Fold Cross Validation accuracy explanation:

- In my scratch model, first I'll divide the datasets into 10 different segments in an interesting way such that the dataset will be divided into 10 exactly equal parts(number of rows), if 500 rows in dataset then 50 each fold_size and these 50 rows will be chose randomly using randrage(dataset_size) function.
- These 10 folds or Arrays of equal size but with different data rows will be used in a different manner. Here's the detailed explanation:

Fold Splitting:

Step 1: The dataset is divided into k folds.
Step 2: In each iteration, one of the k folds is used as the test set, and the remaining k-1 folds are used as the training set.
Training and Testing:
Iteration 1: Fold 1 is the test set, Folds 2 to k are the training set.
Iteration 2: Fold 2 is the test set, Folds 1, 3 to k are the training set.
...
Iteration k: Fold k is the test set, Folds 1 to k-1 are the training set.

- And the evaluate_accuracy methods calculates accuracy based on the simple logic where actual output is matched with the predicted output and ratio of (predicted/actual)*100 is accuracy score, for that fold. This process is repeated for 10 times on above(red highlighted) different folds and we end up getting 10 accuracies, the average accuracy is simply arithmetic mean() of these 10 scores. These accuracies are calculated on both k=3 and k=sqrt(X_train_dataset_Size).


```
*********************** CROSS-VALIDATION RESULTS ***********************

Accuracy Scores(k=3): [38.46153846153847, 38.46153846153847, 23.076923076923077, 38.46153846153847, 46.15384615384615, 61.53846153846154, 38.461538461538
47, 23.076923076923077, 30.76923076923077, 61.53846153846154]

Accuracy Scores (Large k=sqrt(n)): [53.84615384615385, 23.076923076923077, 61.53846153846154, 46.15384615384615, 30.76923076923077, 23.076923076923077, 3
8.46153846153847, 53.84615384615385, 30.76923076923077, 15.384615384615385]

Average Accuracy(k=3): 40.0000
Average Accuracy(k=sqrt(n)): 37.6923
```

2. Scikit-Learn KNN Implementation:

2.1 Dataset Preprocessing:
- Dataset is loaded using Pandas `read_csv` method.
- Columns irrelevant to classification (e.g., name and hobby) are excluded from feature selection.
- Data is split into training and testing sets: `X_train`, `y_train`, `X_test`, and `y_test`.

2.2 Feature Scaling:
- Data is scaled using `StandardScaler()` from Scikit-Learn's preprocessing module.
- Standardization is applied to ensure all features have equal importance during distance calculation.

2.3 Determining k:
- `large_k` is calculated as the square root of the training dataset size, ensuring an odd value.
- An odd k value is chosen to prevent tie-break situations in binary classification.

2.4 KNN Model Initialization:
- Two KNN classifiers are initialized: one with k=3 (`knn`) and another with k=`large_k` (`knn_large_k`).

2.5 Cross-Validation:
- K-Fold Splitting: The training dataset is divided into 10 folds.
- Training and Testing:
- For each fold:
- KNN models are trained on k-1 folds and tested on the remaining fold.

- Cross-validation is performed using Scikit-Learn's `KFold` method.
- Accuracy scores for both k=3 and k=`large_k` are obtained for each fold.
- Accuracy Calculation:
- Accuracy scores are obtained for each fold.
- The `cross_val_score` function from Scikit-Learn computes accuracy using K-Fold cross-validation.
- Results are printed for both k=3 and k=`large_k`.
- Average Accuracy Calculation:
- Average accuracy is calculated for both k=3 and k=`large_k`.
- Average accuracy provides a reliable measure of the model's performance across different folds.

Note: Scikit-Learn's KNN implementation internally handles distance calculations, neighbor selection, and prediction, making the process efficient and accurate. The K-Fold cross-validation ensures a robust evaluation of the model's generalization performance across various subsets of the dataset.

Breast Cancer Dataset

Car Evaluation Dataset - Custom KNN Implementation:

1. **Dataset Loading and Preprocessing:**
   - Loaded the car evaluation dataset using Pandas.
   - Masked missing values in the dataset to ensure data integrity.
   - Converted the dataset to a list for ease of processing.

2. **Feature Encoding:**
   - Converted string values to unique integers for each column using a mapping function.
   - Transformed categorical features into numerical equivalents, making them suitable for KNN classification.

3. **Distance Calculation:**
   - Implemented Euclidean distance calculation to measure similarity between data points.
   - The Euclidean distance formula was applied to relevant columns of the dataset.

4. **Nearest Neighbors Selection:**
   - Developed a function to find k nearest neighbors based on Euclidean distance.
   - Sorted the distances and selected the top k neighbors for prediction.

5. **Classification Prediction:**
   - Utilized the majority voting mechanism to predict the class label.
   - Predicted class was determined by the most common class among the k neighbors.

6. **New Observation Prediction:**
   - Provided a new observation (`new_observation_row`) for prediction.

  - Predicted the class for both k=3 and k=`large_k` using the implemented KNN model.

7. **K-Fold Cross-Validation:**
  - Split the dataset into 10 folds for cross-validation.
  - Developed a function for K-Fold cross-validation to assess model accuracy.
  - Utilized random seed (20) for reproducibility of results.

8. **Results and Evaluation:**
  - Printed accuracy scores for both k=3 and k=`large_k` for each fold.
  - Calculated and displayed the average accuracy for both k values.

Note: The custom KNN implementation showcases the process of dataset preprocessing, distance calculation, neighbor selection, and classification prediction. K-Fold cross-validation ensures robust evaluation, providing insights into the model's performance across different subsets of the dataset.

Breast Cancer Dataset - Scikit KNN Implementation:

1. **Data Preprocessing:**
  - Extracted features and target variable from the dataset.
  - Standardized features using `StandardScaler` to bring them to the same scale.

2. **Choosing K-Values:**
  - Determined a suitable large k-value based on the square root of the dataset size.
  - Ensured that `large_k` was even for an optimal KNN model.

3. **K-Fold Cross-Validation:**
  - Utilized `KFold` from Scikit-learn to perform 10-fold cross-validation.
  - Applied random seed (20) for reproducible results.
  - Implemented KNN classifiers with k=3 and k=`large_k`.

4. **Model Evaluation:**
  - Calculated accuracy scores for both k=3 and k=`large_k` using `cross_val_score`.
  - Presented accuracy scores for each fold for both k-values.
  - Computed average accuracy for both k-values and displayed in percentage format.

Note: The Scikit-learn implementation employs K-Fold cross-validation to validate the KNN model's performance on the breast cancer dataset. The use of standardized features ensures that each feature contributes equally to the distance calculation, enhancing the model's accuracy.

Car Evaluation Dataset

Car Evaluation Dataset - KNN Implementation:

1. **Data Preprocessing:**
   - Loaded the car evaluation dataset and handled missing values by masking them.
   - Converted string values to unique integer representations for each column using a mapping strategy.

2. **KNN Implementation:**
   - Defined a function to calculate Euclidean distance between two rows.
   - Implemented functions to find k-nearest neighbors and predict the class based on majority voting.

3. **New Observation Prediction:**
   - Created a new observation row with specific feature values.
   - Determined a suitable large k-value based on the square root of the dataset size.
   - Made predictions for the new observation using k=3 and k=`large_k`.

4. **Cross-Validation and Evaluation:**
   - Implemented a function to split the dataset into 10 folds for cross-validation.
   - Utilized the KNN model to make predictions for each fold.
   - Calculated accuracy scores for both k=3 and k=`large_k` for each fold and presented them.
   - Computed average accuracy for both k-values and displayed the results.

Note: The KNN algorithm was applied to the car evaluation dataset, enabling predictions for new observations and evaluating its performance through 10-fold cross-validation. The choice of `k` significantly impacts prediction accuracy, with k=3 representing a smaller, localized influence, and k=`large_k` offering a broader perspective.

Breast Cancer Dataset - KNN Implementation:

1. **Data Preprocessing:**
   - Extracted features and target variable from the dataset.
   - Applied feature scaling using StandardScaler to standardize feature values.
   - Determined a suitable large k-value based on the square root of the dataset size.

2. **KNN Model Initialization:**
   - Initialized KNeighborsClassifier with k=3 for the localized influence of neighbors.
   - Initialized another KNeighborsClassifier with a large k-value to have a broader perspective.

3. **Cross-Validation and Evaluation:**
   - Utilized KFold with 10 splits and shuffling for cross-validation.
   - Applied the KNN model with k=3 and `large_k` to make predictions for each fold.
   - Calculated accuracy scores for both k=3 and k=`large_k` for each fold and presented them.

  - Computed average accuracy for both k-values and displayed the results.

Note: The KNN algorithm was implemented on the breast cancer dataset, providing accuracy scores for both k=3 and a larger k-value. Utilizing 10-fold cross-validation ensures robust evaluation of the model's performance across different data subsets.

Paired T-Hypothesis Testing

Choice of Paired T-Test for Comparison:

The choice of a statistical test depends on the nature of the data and the research question being addressed. In this case, a paired t-test was selected for several reasons:

1. **Paired Data:** The accuracy scores being compared (from the scratch implementation and scikit-learn implementation) are paired data points for the same datasets. Each accuracy score from the scratch implementation corresponds to an accuracy score from the scikit-learn implementation for the same dataset. This pairing is essential because it controls for individual dataset differences.

2. **Comparison of Means:** The objective is to compare the mean accuracy scores between the two implementations for each dataset. The paired t-test is suitable for comparing the means of two related groups. It assesses whether there is a significant difference between the means of the paired observations.

3. **Normality Assumption:** The paired t-test assumes that the differences between the paired observations are normally distributed. While this assumption is crucial, it is often robust to violations, especially with larger sample sizes. Given that accuracy scores are percentages and tend to be distributed between 0 and 100, the normality assumption is reasonable, particularly with the larger number of folds used for cross-validation (10 folds in this case).

4. **Effect of Outliers:** The paired t-test is relatively robust against outliers. Outliers might occur in real-world datasets due to various reasons. Robustness against outliers is essential for the analysis to be reliable.

5. **Matched Samples:** The paired t-test is appropriate for matched or paired samples, ensuring that the comparison is made under similar conditions for both implementations.

Considering these factors, the paired t-test was an appropriate choice for comparing the accuracy scores between the KNN implementations. It allowed for a rigorous statistical evaluation of whether there was a significant difference in accuracy between the two methods, providing valuable insights into the comparative performance of the scratch implementation and the scikit-learn implementation.

Interpretations of Each dataset conclusion using Paired T-Tesing:

Statistical Analysis and Conclusion:

In the conducted analysis, paired t-tests were performed to compare the accuracy scores between the KNN implementations from scratch and using scikit-learn for three different datasets: Car Evaluation, Breast Cancer, and Hayes Roth. The aim was to assess if there is a significant difference in accuracies between the two implementations.

Car Evaluation Dataset:

T-statistic: -6.5065

P-value: 0.0001

Conclusion: The null hypothesis (H0) was rejected for the Car Evaluation dataset, indicating a significant difference in accuracies between the KNN implementations. The scikit-learn implementation outperformed the scratch implementation significantly for this dataset.

Breast Cancer Dataset:

T-statistic: -0.2138

P-value: 0.8355

Conclusion: The null hypothesis (H0) was not rejected for the Breast Cancer dataset, suggesting no significant difference in accuracies between the KNN implementations.

Hayes Roth Dataset:

T-statistic: N/A

P-value: N/A

Conclusion: The t-test could not be performed for the Hayes Roth dataset due to calculation issues, requiring further investigation.

Overall Conclusion:

For the Car Evaluation dataset, the scikit-learn implementation of KNN significantly outperformed the scratch implementation, indicating its superiority.

For the Breast Cancer dataset, both implementations yielded comparable results, suggesting their interchangeability.

Further investigation is needed to resolve issues with the Hayes Roth dataset before drawing any conclusions.

Recommendation:

It is recommended to utilize the scikit-learn implementation of KNN for the Car Evaluation dataset due to its significantly higher accuracy. For the Breast Cancer dataset, both implementations can be considered interchangeable. Further investigation is required to resolve issues with the Hayes Roth dataset before making any decisions.

REFERENCES:

1. Original Link by professor:
   - ➔ https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
2. Youtube Link:
   - ➔ https://www.youtube.com/watch?v=4HKqjENq9OU&t=319s
3. How and why to fillna(NaN), instead of empty values:
   - ➔ https://sparkbyexamples.com/pandas/pandas-replace-blank-values-with-nan/?expand_article=1
4. Some implementation of code lines from this good article:
   - ➔ https://www.linkedin.com/pulse/k-nearest-neighbor-algorithm-from-scratchwithout-library-ritika-yadav/
5. What is K-Fold and how to implement using Scikit:
   - ➔ https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn
6. Choosing right k-value(sqrt(dataset_size)):
   - ➔ https://stackoverflow.com/questions/11568897/value-of-k-in-k-nearest-neighbor-algorithm

Regarding choosing Paired T-testing and how to implement it:

7. https://www.analyticsvidhya.com/blog/2021/09/hypothesis-testing-in-machine-learning-everything-you-need-to-know/
8. https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89e169ce
9. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html
10. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html