Final Term Project

Data Mining

CS- 634-104

FALL-21

Professor: Yasser Abduallah

Department of Computer Science

NAME: Kavya Basavalingappa Umashankar

UCID – kbu2

GITHUB LINK

Contents **Pg no**

Introduction:

Algorithm Overview:

Computers that learn from data and use algorithms to execute tasks without being explicitly programmed are referred to as machine learning. Deep learning is based on a complicated set of algorithms that are modeled after the human brain. This allows unstructured data, such as documents, photos, and text, to be processed.

K Nearest Neighbours:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

Random Forest Algorithm:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set.

Steps involved in Random forests :

- **Step-1:**Randomly pick $\sqrt{total\ number\ of\ attributes}$
- **Step-2:**Calculate the gini impurity
- **Step-3:**Split based on gini impurity

- **Step-4:**Repeat the above process for different attributes
- **Step-5:**Consider the highest gini impurity and build 1 random tree
- **Step-6:**Repeat this process to form N number of trees.

Long Short Term Memory:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

K- FOLD CROSS VALIDATION:

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    a. Take the group as a hold out or test data set
    b. Take the remaining groups as a training data set
    c. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model

Summarize the skill of the model using the sample of model evaluation scores

ASSUMPTIONS:

The implementation has some assumptions to make sure the running of the application is flawless and without exception.

- Dataset sit cleaned.
- Dataset is not imbalance.
- All the attributes have correlation with the output variable

REQUIREMENTS:

- Modern Operating System:
- Windows 7, 8 or 10 / Mac OS X 10.11 or higher, 64-bit / Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
- x86 64-bit CPU (Intel / AMD architecture)
- 4 GB RAM
- 5 GB free disk space
- Python IDE: PyCharm/Spyder with Numpy and Pandas - Reading and usage of data,sklearn, keras and tensorflow- for models, matplotlib and Seaborn -for heatmap installed.

DATASET OVERVIEW:

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. I have selected the redwine variants. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available.This data set was

prepossessed and downloaded from the UCI Machine Learning Repository. This data set was simple, cleaned, practice data set for classification modelling.

Attribute Information:

Input variables (based on physicochemical tests):

1 - fixed acidity (float values in range of 4.600 to 15.900)

2 - volatile acidity (float values in range of 0.12 to 1.58)

3 - citric acid (Float values in range of 0 to 1)

4 - residual sugar (float values in range of 0.9 to 15.5)

5 - chlorides (float values in range of 0.012 to 0.6110)

6 - free sulfur dioxide (integer values in range of 1 to 72)

7 - total sulfur dioxide (float values in range of 6 to 289)

8 - density (float values in range of 0.990 to 1.000)

9 - pH (float values in range of 2.74 to 4.01)

10 - sulphates (float values in range of 0.33 to 2.000)

11 - alcohol (float values in range of 8.40 to 14.90)

Output variable (based on sensory data):

12 - quality ('good' or 'bad')

COMPILE AND RUN THE SOURCE CODE:

Python automatically compiles the code before running it. The file is saved as "Evaluation.py" and all required files are present in the directory. In the terminal, the command **python Evaluation.py** in pycharm command prompt and **runfile("address of the file", wfile="address of execution file of sypder")** in spyder will compile and execute the code. All the assumptions are in place as explained in the "Assumptions" Section.

TESTING IMPLEMENTATION:

K Fold Cross Validation:

Implementing K Fold Cross validation manually for K Nearest Neighbours, Random Forest, Long Short Term Memory.

Out[11]:

fold	model	Confusion matrix	TruePositive rate	True Negative rate	False positive rate	False Negative rate	Accuracy	Precision	Recall	F1 measure	Error rate	Balance Accuracy	TSS	HSS
1	KNN	[[106, 33], [7, 14]]	0.297872	0.938053	0.061947	0.702128	0.750000	0.666667	0.297872	0.411765	0.250000	0.617963	0.235925	0.281383
1	RF	[[95, 23], [18, 24]]	0.510638	0.840708	0.159292	0.489362	0.743750	0.571429	0.510638	0.539326	0.256250	0.675673	0.351346	0.362612
1	LSTM	[[0, 0], [113, 47]]	1.000000	0.000000	1.000000	0.000000	0.293750	0.293750	1.000000	0.454106	0.706250	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
2	KNN	[[72, 53], [10, 25]]	0.320513	0.878049	0.121951	0.679487	0.606250	0.714286	0.320513	0.442478	0.393750	0.599281	0.198562	0.201268
2	RF	[[65, 33], [17, 45]]	0.576923	0.792683	0.207317	0.423077	0.687500	0.725806	0.576923	0.642857	0.312500	0.684803	0.369606	0.371464
2	LSTM	[[0, 0], [82, 78]]	1.000000	0.000000	1.000000	0.000000	0.487500	0.487500	1.000000	0.655462	0.512500	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
3	KNN	[[40, 59], [15, 46]]	0.438095	0.727273	0.272727	0.561905	0.537500	0.754098	0.438095	0.554217	0.462500	0.582684	0.165368	0.138909
3	RF	[[37, 21], [18, 84]]	0.800000	0.672727	0.327273	0.200000	0.756250	0.823529	0.800000	0.811594	0.243750	0.736364	0.472727	0.466667
3	LSTM	[[0, 0], [55, 105]]	1.000000	0.000000	1.000000	0.000000	0.656250	0.656250	1.000000	0.792453	0.343750	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
4	KNN	[[43, 50], [26, 41]]	0.450549	0.623188	0.376812	0.549451	0.525000	0.611940	0.450549	0.518987	0.475000	0.536869	0.073738	0.070763
4	RF	[[49, 22], [20, 69]]	0.758242	0.710145	0.289855	0.241758	0.737500	0.775281	0.758242	0.766667	0.262500	0.734193	0.468387	0.466751
4	LSTM	[[0, 0], [69, 91]]	1.000000	0.000000	1.000000	0.000000	0.568750	0.568750	1.000000	0.725100	0.431250	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
5	KNN	[[85, 39], [21, 15]]	0.277778	0.801887	0.198113	0.722222	0.625000	0.416667	0.277778	0.333333	0.375000	0.539832	0.079665	0.086758
5	RF	[[92, 33], [14, 21]]	0.388889	0.867925	0.132075	0.611111	0.706250	0.600000	0.388889	0.471910	0.293750	0.628407	0.256813	0.281071
5	LSTM	[[0, 0], [106, 54]]	1.000000	0.000000	1.000000	0.000000	0.337500	0.337500	1.000000	0.504673	0.662500	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
6	KNN	[[39, 45], [24, 52]]	0.536082	0.619048	0.380952	0.463918	0.568750	0.684211	0.536082	0.601156	0.431250	0.577565	0.155130	0.146568
6	RF	[[36, 12], [27, 85]]	0.876289	0.571429	0.428571	0.123711	0.756250	0.758929	0.876289	0.813397	0.243750	0.723859	0.447717	0.467213
6	LSTM	[[0, 0], [63, 97]]	1.000000	0.000000	1.000000	0.000000	0.606250	0.606250	1.000000	0.754864	0.393750	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
7	KNN	[[34, 56], [9, 61]]	0.521368	0.790698	0.209302	0.478632	0.593750	0.871429	0.521368	0.652406	0.406250	0.656033	0.312065	0.231905
7	RF	[[24, 21], [19, 96]]	0.820513	0.558140	0.441860	0.179487	0.750000	0.834783	0.820513	0.827586	0.250000	0.689326	0.378652	0.373164
7	LSTM	[[0, 0], [43, 117]]	1.000000	0.000000	1.000000	0.000000	0.731250	0.731250	1.000000	0.844765	0.268750	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
8	KNN	[[36, 52], [18, 54]]	0.509434	0.666667	0.333333	0.490566	0.562500	0.750000	0.509434	0.606742	0.437500	0.588050	0.176101	0.152542
8	RF	[[35, 18], [19, 88]]	0.830189	0.648148	0.351852	0.169811	0.768750	0.822430	0.830189	0.826291	0.231250	0.739168	0.478337	0.480519
8	LSTM	[[0, 0], [54, 106]]	1.000000	0.000000	1.000000	0.000000	0.662500	0.662500	1.000000	0.796992	0.337500	0.500000	0.000000	0.000000

nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
9	KNN	[[60, 53], [25, 22]]	0.293333	0.705882	0.294118	0.706667	0.512500	0.468085	0.293333	0.360656	0.487500	0.499608	-0.000784	-0.000802
9	RF	[[64, 19], [21, 56]]	0.746667	0.752941	0.247059	0.253333	0.750000	0.727273	0.746667	0.736842	0.250000	0.749804	0.499608	0.498825
9	LSTM	[[0, 0], [85, 75]]	1.000000	0.000000	1.000000	0.000000	0.468750	0.468750	1.000000	0.638298	0.531250	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
10	KNN	[[48, 44], [26, 41]]	0.482353	0.648649	0.351351	0.517647	0.559748	0.611940	0.482353	0.539474	0.440252	0.565501	0.131002	0.128972
10	RF	[[41, 16], [33, 69]]	0.811765	0.554054	0.445946	0.188235	0.691824	0.676471	0.811765	0.737968	0.308176	0.682909	0.365819	0.371339
10	LSTM	[[0, 0], [74, 85]]	1.000000	0.000000	1.000000	0.000000	0.534591	0.534591	1.000000	0.696721	0.465409	0.500000	0.000000	0.000000
nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Figure 1: K Fold Cross validation with metrics for each fold : K Nearest Neighbours, Random Forest, Long Short Term Memory.

Out[12]:

	model	AVG TruePositive rate	AVG True Negative rate	AVG False positive rate	AVG False Negative rate	AVG Accuracy	AVG Precision	AVG Recall	AVG F1 measure	AVG Error rate	AVG Balance Accuracy	AVG TSS	AVG HSS
0	KNN	0.412738	0.739939	0.260061	0.587262	0.584100	0.654932	0.412738	0.502121	0.415900	0.576339	0.152677	0.143827
1	RF	0.712011	0.696890	0.303110	0.287989	0.734807	0.731593	0.712011	0.717444	0.265193	0.704451	0.408901	0.413963
2	LSTM	1.000000	0.000000	1.000000	0.000000	0.534709	0.534709	1.000000	0.686343	0.465291	0.500000	0.000000	0.000000

Figure 2: Average of all the metrics calculated at each fold of K Fold Cross validation.

RESULTS:

The Average metrics of the 3 models are shown in figure 2. The accuracy of KNN is 58.41% and the accuracy of Random Forest is73.48% and the accuracy of Long Short term Memory is 53.47%. The balanced accuracy of KNN is 57.63% and the balanced accuracy of Random Forest is 70.44% and the balanced accuracy of Long Short term Memory is 50.00%.

Random forest algorithm performs better because it creates n number of trees by choosing random attributes, it chooses the class which is provided max output from the trees. From the above observations we can conclude that Random Forest performs better.

REFERENCES

1. https://www.kaggle.com/nareshbhat/wine-quality-binary-classification -Dataset
2. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html -KNN classifier
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html - Random forest Classifier
4. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ -LSTM
5. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM -LSTM
6. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ - LSTM