APPLICATION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN PYTHON PROGRAM LANGUAGE

Aleksandar Miljković[1]
Ministry of the Interior of the Republic of Serbia

Slobodan Nedeljković
Ministry of the Interior of the Republic of Serbia

Milan Čabarkapa, PhD
School of Electrical Engineering, University of Belgrade, Serbia

Marko Vuković
Dragan Randjelović, PhD
Faculty of Diplomacy and Security, University Union – Nikola Tesla, Belgrade, Serbia

Vladica Stojanović, PhD
University of Criminal Investigation and Police Studies, Belgrade, Serbia

Abstract: This paper overviews the practical application of the Python programming language in the artificial intelligence (AI) and machine learning (ML) field. The possibilities and advantages, as well as the current challenges of Python programming language, are summarized. Moreover, the capabilities of the considered technologies are demonstrated in the practical examples. Although the provided examples are not very complex, they can clearly indicate the principles of using Python programming language in the AI and ML fields for solving statistically-related problems of clustering and classification. This work and the results presented in this paper can be used as a helpful guide and basis for advanced implementation based on Python programing language for solving different practical AL and ML problems in different sciences, government and industry areas.

Keywords: artificial intelligence, Python language application, machine learning, software development

1 aleksandar.miljkovic@mup.gov.rs

INTRODUCTION

The ability of computers to perform various tasks has experienced exponential growth. The programming capabilities and performances of computer systems have been significantly improved regarding work domains, processing speed, and size reduction. As a part of the computer science field, artificial intelligence (AI) aims to program computers or machines to be as intelligent as human beings. In other words, the main aim of artificial intelligence is to develop programs that will enable computer, a computer-controlled robot, or software to think intellectually, in a similar way as humans. AI is developed by studying how the human brain thinks and how people learn, decide, and work while trying to solve the problem. Due to the strong computing power of computer systems, the question of if a machine can think and behave like a human being has appeared. Thus, the development of AI began with the intention of creating machines possessing intelligence similar to that of humans.

The AI applications can be developed using various programming languages. However, according to the results presented in recent studies related to the application of different programming languages into the AL filed, the best overall performance has Python programming language. The detailed analysis of the Python language capabilities is presented in the recent work of Nagpal and Gabrani (Nagpal & Gabrani, 2019) and it will be omitted in this work. However, they have neither provided any illustrative examples of the application of the Python programming language in the classification application nor the procedure of classification performance evaluation, which is the main objective of this work. To the best of authors' knowledge, there has been no study illustrating the practical application of Python programming language implementation in AI-based applications, specifically the classification applications based on the machine learning algorithms.

The rest of the paper is organized as follows. The necessity for studying AI is explained in the second section. The fundamental AI-related terminology is presented in the third section. The reasons for Python implementation in the AI field are provided in the fourth section. Machine learning and its fundamental algorithms are introduced in the fifth section. The metrics and methods for evaluation of classification performance are presented in the sixth section. The demonstrative examples of Python implementation in machine learning applications are given in the seventh section. Finally, conclusions are derived in the eighth section.

NECESSITY FOR STUDYING AI

As mentioned above, AI aims to create machines as intelligent as humans. Therefore, there are several reasons why it is important to study AI, and they are listed and explained in the following.

1. *AI can learn from data*

In our everyday life, people are dealing with a huge amount of data, and the human brain cannot process all the information, which is why it is necessary to automate things. AI is suitable for automation because it can learn from data and perform tasks that are repeated with high accuracy and without fatigue.

2. *AI can teach itself*

It is very important that a system can teach itself because data are constantly changing, so knowledge derived from such data must be constantly updated. AI can be used to achieve this purpose because an AI-based system can teach itself.

3. *AI can respond in real time*

Artificial intelligence, with the help of neural networks, can deepen data analysis, which enables AI to think and respond to situations in real time.

4. *AI achieves high precision*

With the help of deep neural networks, AI can achieve very high precision. This feature has promoted the AI application in diagnosing diseases such as cancer.

5. *AI can organize data so as to obtain the best results*

Data denote an intellectual property for systems using self-learning algorithms. AI is needed for indexing and organizing data in a way to achieve the best possible results.

6. *Understanding intelligence*

By using AI, intelligent systems can be built. It is needed to understand the concept of intelligence that the human brain can build another intelligent system like itself.

FUNDAMENTAL AI-RELATED TERMS

Artificial intelligence represents a wide research field of study, and it can help to find solutions to the real-world problems.

Machine learning

Machine learning (ML) is one of the most popular AI fields. The basic concept of machine learning is to make machine learns from data in the same way the humans learn from their own experience. It contains learning models based on which predictions about unknown data can be made.

Logic

This is another important research field in which mathematical logic is used to execute computer programs. It contains rules and facts for conclusions, semantic analysis, etc.

Search

This research field is basically used in games such as chess and tic-tac-toe. Search algorithms provide an optimal solution after searching the entire search space.

Artificial neural networks (ANNs)

ANNs represent parallel computing systems, and they have been developed based on the analogy with biological neural networks. ANNs have been used in robotics, speech recognition, speech processing, and many other fields.

Genetic algorithm

The genetic algorithm helps to solve problems with the help of more than one program, where the final result represents the most suitable result among all the obtained results.

Knowledge representation

Knowledge representation is used to present facts and information in a way that is understandable to machines. The more efficient the knowledge is, the more intelligent the system will be.

PYTHON APPLICATION IN AI FIELD

Due to excellent performances, AI represents the future technology. There have been many AI-based applications presented in the literature. Also, AI has attracted great attention from both industry and academia. However, the main question that arises is in which program language AI-based applications should be developed to achieve the best possible performances. Many program languages, including Lisp, Prolog, C++, Java, and Python, have been used to develop AI applications. Among them, Python program language has been gaining popularity recently, due to the following reasons.

Simple syntax and less programming in Python

Python provides developing applications with less coding and simpler syntax than the other programming languages that are beneficial to AI application development, making testing easier, and allowing developers to focus more on programming.

Built-in libraries for AI projects

The main advantage of using Python in AI is that it comes with embedded libraries. Python has libraries for almost all types of AI projects, including the NumPy, SciPy, matplotlib, nltk, SimpleAI (NumPy documentation, 2020), (SkitLearn documentation, 2020), (Python documentation, 2020).

Also, Python is the open-source programming language, so it can be used in a wide range of software tasks, from small shell scripts to advanced corporate web applications (Nanz & Furia, 2015).

MACHINE LEARNING AS MAIN AI METHODOLOGY

Learning denotes the acquisition of knowledge or skills through study or experience. On this basis, ML can be considered as a part of the computer science field that provides computer systems with the ability to learn from experience without being explicitly programmed. The main focus of machine learning is to enable computers to learn without human intervention. Thus, ML algorithms help the computer system learn without explicit programming. The ML algorithms can be roughly divided into supervised and unsupervised (Shalev-Shwartzm & Ben-David, 2014).

Supervised ML algorithms

These algorithms are the most commonly used machine learning algorithms. They are called supervised because the learning process is supervised. In supervised ML algorithms, possible outcomes are already known, and the training data are indicated by the exact answers. There are two types of supervised learning problems, classification and regression. The most common supervised ML algorithms are the decision trees, random forest, *k*-nearest neighbour algorithm, and logistic regression.

Unsupervised ML algorithms

These machine learning algorithms are not supervised. Non-supervised learning problems can be divided into two types of problems, clustering and association. The *k*-means clustering and a priori algorithm are common unsupervised machine learning algorithms.

Most frequent machine learning algorithms

Linear regression

This is one of the most popular algorithms in statistics and machine learning. This is a linear model that assumes a linear relationship between the input and output variables. There are two types of linear regression, simple linear regression which represents the linear regression that has only one independent variable, and multiple linear regression which is linear regression having more than one independent variable.

Linear regression is mainly used to estimate the real values based on a continuous variable, one or more of them. For example, the total sales of a store in one day, based on real values, can be estimated by linear regression (Haslwanter, 2016).

Logistic regression

Logistic regression is a classification algorithm used to estimate discrete values, such as zero or one, true or false, whether or not, based on a given set of independent variables. Basically, it predicts probability; therefore, its output is in the interval between zero and one (Shalev-Shwartzm & Ben-David, 2014).

Decision tree

The decision tree is a supervised learning algorithm mainly used for classification problems. The decision tree has nodes that form a rooted tree that is a directed tree with a node called "root". Root has no incoming branches, and all other nodes have one input branch. These nodes are called leaves or the deciding nodes.

Support Vector Machine (SVM)

It is mainly used for classification problems. The main concept of SVM is to draw each data item as a point in an *n*-dimensional space, with the value of each property being a value of a certain coordinate; the classifier represents the line border between the data classes, as shown in Figure 1.

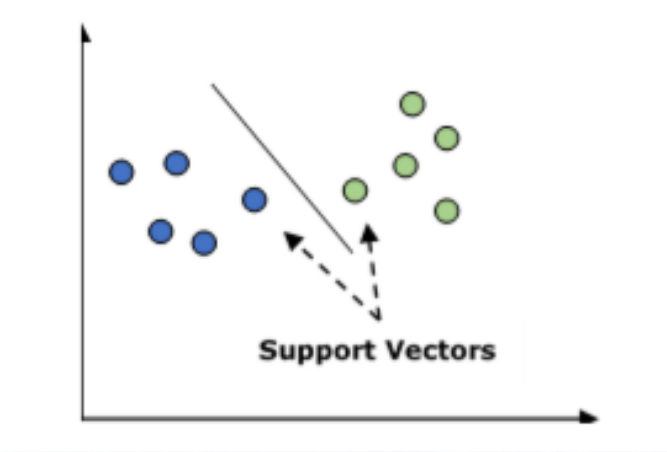

Figure 1: Support Vector Algorithm

Naive Bayes

This is a classification technique that uses the Bayesian theorem for constructing a classifier. The assumption is that the predictors are independent, i.e. it is assumed that the presence of a particular property in a class is not related to the presence of any other property. The Naive Bayes model is easy to build and particularly useful for large datasets (Haslwanter, 2016).

k-Nearest Neighbours (kNN)

It is widely used to solve the classification problem. The basic concept of this algorithm is that it is used to store all available cases and classifies new cases by the majority of its neighbour's votes. The case is then assigned to the most common class among its closest neighbours, measured by the distance function. The distance function can be Euclidean, Minkowski, and Hamming distance functions. The *k*KNN has higher computational complexity than the other classification algorithms. Also, data normalization is required; otherwise, the higher-range variables can be biased, and data preprocessing is also needed to remove the noise (Shalev-Shwartzm & Ben-David, 2014)**.**

k-Means Clustering

This algorithm is used to solve cluster problems. Basically, this algorithm is unsupervised, and it divides the dataset into a certain number of clusters, which are formed by the *k*-means process, where *k* denotes the number of points in each cluster. The algorithm's centroids of each cluster are based on the existing cluster members. The clustering is performed until the algorithm converges.

Random Forest

This is a supervised classification algorithm that can be used for both classification and regression type problems. Basically, it represents a collection of decision trees, i.e. forests. This algorithm can handle the missing values.

CLASSIFICATION PERFORMANCE MEASURING

After the machine learning algorithm development, its effectiveness was evaluated using different performance metrics. The choice of the evaluation metric is very important because the choice of a metric affects how the measurement and comparison of the performance of the machine learning algorithm are performed. The criteria used in the evaluation process are given in the following.

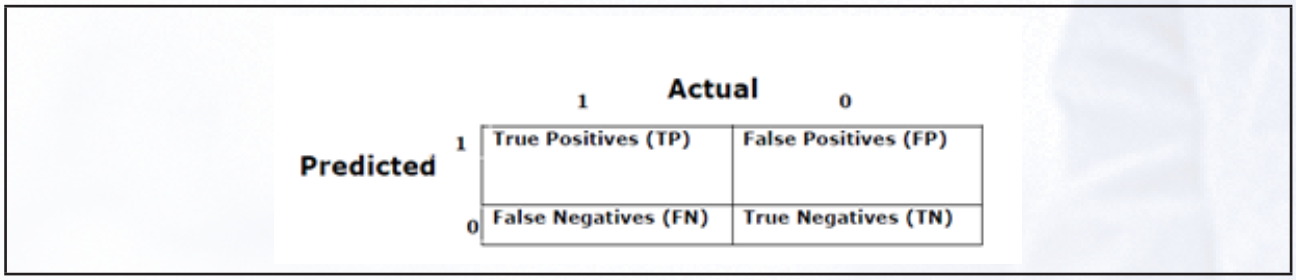

Figure 2: Confusion matrix

Confusion matrix – This metric is commonly used in classification problems where the output can be of two or more class types. Also, using this metric is the easiest way to measure the performance of a classifier. The confusion matrix is essentially a table with two dimensions, which are "Real" and

"Predicted". Both dimensions have True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN) fields.

In the confusion matrix, one is used to denote the positive class, and zero is used to denote the negative class. The following terms are related to the confusion matrix:

True Positives - cases where the real data class is one, and one is predicted.

True Negatives - cases where the actual data point class is zero, and one is predicted.

False Positives - cases where the actual data class is zero, and one is predicted.

False Negatives - cases where the actual data point class is one, and zero is predicted.

Accuracy - The confusion matrix itself is not a measure of performance as such, but almost all performance matrices are based on a confusion matrix. One of them is accuracy. In the classification problem, accuracy can be defined as the number of correct predictions made by a model over all types of prediction, and it is calculated as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

Precision - Precision shows consistent results when measurements are repeated, and it is calculated by:

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

Recall or Sensitivity - It can be defined as a number of positive models that the model retrieves, and it is calculated as:

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

Specificity - It can be defined as a number of negative model retrievals, and it is calculated as:

$$Specificity = \frac{TN}{TN+FP} \quad (4)$$

Real-world data are not naturally organized into a certain number of characteristic clusters, so it is not easy to draw conclusions. Therefore, it is necessary to evaluate clustering performance as well as its quality, which can be done using the silhouette analysis.

Silhouette Analysis

This method evaluates the quality of clustering by measuring the distance between clusters. Basically, it provides a way to evaluate parameters such as the number of clusters by giving a silhouette. The result is a metric that shows how close each point in a cluster is to the points in adjacent clusters.

Analysis of silhouette results

The result of the silhouette analysis is in the range of [-1, 1], where +1 means that a sample is far from the adjacent cluster, 0 means that a sample is at or near the decision boundary between two adjacent clusters, and -1 means that a sample is assigned to the wrong cluster.

DEMONSTRATIVE EXAMPLES OF PYTHON AI IMPLEMENTATION

Supervised learning: classification

The classification technique aims to obtain conclusions from the observed values. In the classification problem, there is a categorized output, such as "black" or "white" or "teaching" and "non-teaching". When building a classification model, it is necessary to have a training dataset that contains data points and the corresponding tags. Classification models are mainly used in face recognition, spam identification, and many other computer vision-related applications (James, 2017)

Python-based classier

In order to build a classifier in Python, Python 3 and Scikit-learn were used as a tool for machine learning. The building process included three following steps (James, Witten, Hastie & Tibshirani, 2017).

Step 1: *Import Scikit-learn*

In this step, the Python package called Scikit-learn, which is one of the best Python Machine Learning Modules, was installed by the following command:

Import Sklearn

Step 2: *Importing Scikit-learn dataset*

In this step, the dataset used to develop a machine learning model was obtained. The Diagnostic Breast Cancer Basis in Wisconsin was used. The dataset included different information on tumours of breast cancer, as well as the classification codes for malignant or benign. The dataset had 569 copies of 569 tumours and included the information about 30 attributes or traits, such as tumour radius, texture, smoothness, and area. The Scikit-Learned Breast Cancer dataset was imported by the following command:

from sklearn.datasets import load_breast_cancer

The data were loaded using the command:

data = load_breast_cancer()

A list of important vocabulary keys included the following attributes:

- Classification target names (target_names)
- Real targets (target)
- Names of features/traits (feature_names)

The loaded dataset was organized by the following commands:

label_names = data['target_names']

```
labels = data['target']
feature_names = data['feature_names']
features = data['data']
```

Next, the class labels that included the first instance of the data, our property names, and the value of the function, were printed using the following command:

```
print(label_names)
```

Step 3: *Organizing dataset into subsets*

In this step, the original dataset was divided into two subsets, training set and test set, without any data overlapping between the sets. The dataset was divided by the function *train_test_split()* using the following commands:

```
from sklearn.model_selection import train_test_split
train, test, train_labels, test_labels = train_test_split(features,labels,test_size = 0.40, random_state = 42)
```

The second command in the above coded was used to divide the data in that way that the test data represented 40% of the overall data, and the remaining data were used for model training.

Step 4: *Model building*

The Naive Bayesian algorithm was used for model building running the following commands:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()     # command for model initialization
model = gnb.fit(train, train_labels)     # training data adaptation to the model
```

Step 5: *Model accuracy evaluation*

The model was evaluated by anticipating the test data using the following commands:

```
preds = gnb.predict(test)
print(preds)
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1
 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0
 0 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 0 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 0 1]
```

The above series of zeros and ones represent the predicted values of the tumour classes - malignant and benign.

Model accuracy was evaluated by the following commands:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(test_labels,preds))
0.951754385965
```

The result showed that the Naive Bayesian classifier achieved an accuracy of 95.17%.
The entire Python classifier code is provided below.

```
import sklearn

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# loads dataset
# in this example, a set of data to be classified is loaded
# this dataset is an example from the sklearn library and is related to
data = load_breast_cancer()

# from the set of data, the groups in which data are to be classified are extracted
label_names = data['target_names']

# the values are extracted, related to which element of the list belongs to which group
labels = data['target']

# the names for the characteristics describing each of the data are extracted
feature_names = data['feature_names']

# the values for the characteristics describing each of the data are extracted
features = data['data']

# the dataset is divided into two groups
# one group is used for training
# another is to test the performance of the model
#to split dataset, the treain_test_split method is used to which the data is transmitted and to which group belongs
#defined test_size defines which part of the data to use for testing, while the rest of the data will be used for training
#random_state variable is determined based on which values to randomly select values
train, test, train_labels, test_labels = train_test_split(features,labels,test_size = 0.40, random_state = 42)

# a new GaussianNB class object is instantiated
# which will be used for classification
gnb = GaussianNB()

# the fit method creates a new model based on the set of data specified for training
gnb.fit(train, train_labels)

#by using the predict method, the classification of the test data is executed based on #the defined model
preds = gnb.predict(test)

# by using the accuracy_score method, we compare real values with the test ones
#in this way, we determine how well the trained model is
# we print this value on the screen
print(accuracy_score(test_labels,preds))
```

Unsupervised learning: clusterization

Unsupervised machine learning algorithms do not have a supervisor that would provide guidance. In learning without supervision, there is no correct answer and teacher for guidance, so algorithms themselves need to discover an interesting form in learning data (Hugunin, 1999).

Clustering is a type of unsupervised method and a common technique for statistical data analysis used in many areas. Grouping is usually the task of dividing a set of observations into subgroups, called clusters, in such a way that observations in the same cluster are similar while those in different clusters are different (Pandas documentation, 2020). Thus, the basic goal of clustering is to group data based on similarity and diversity, as shown in Figure 3.

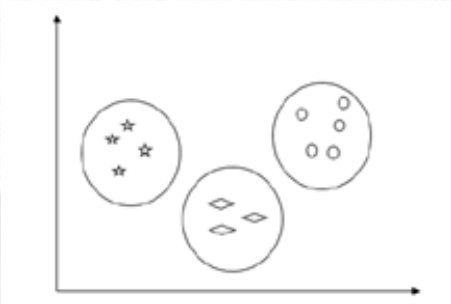

Figure 3: Illustration of data clustering into three clusters

Clustering using *k*-means algorithm

The *k*-means method is one of the well-known algorithms for data grouping. In this work, the straight clustering is considered, where the number of clusters is defined at the beginning of the algorithm. This is iterative grouping logarithm, which includes the following steps:

Step 1: *Define the number of clusters denoted as k.*

Step 2: *Randomly assign each data sample to one of the clusters*

In this step, centroid clusters are calculated. Since this is an iterative algorithm, *k* centroid locations are updated in each iteration until a global optimum is found. The *k*-means algorithm for grouping developed using Python programing language adopting the Scikit-learn module is given in the following (NumPy documentation, 2020), (SkitLearn documentation, 2020).

 # Intorduction of necessary packages:

```
import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

import numpy as np

from sklearn.cluster import Kmeans
```

 # Generation of a two-dimensional dataset that contains four blobs, using make_blob from the sklearn. dataset package

```
from sklearn.datasets.samples_generator import make_blobs
```

```
X, y_true = make_blobs(n_samples=500, centers=4,
cluster_std=0.40, random_state=0)
# Dataset clustering visualization—refer to Figure 4
plt.scatter(X[:, 0],  X[:, 1],  s=50);
plt.show()
```

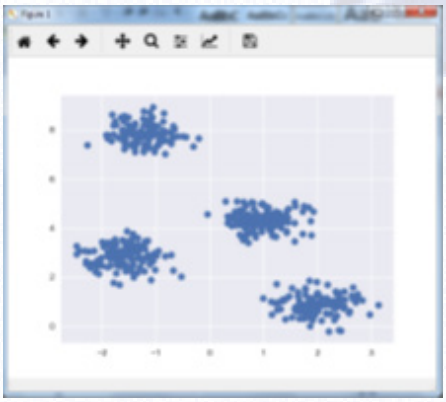

Figure 4: Visualization of the dataset clustering process

The *k*-means object with a defined number of clusters was generated by the following command:

kmeans = KMeans(n_clusters=4)

The number of clusters was equal to four. Next, the training process was perfomed by the following commands:

```
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
# Dataset clustering visualization—refer to Figure 5
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
plt.show()
```

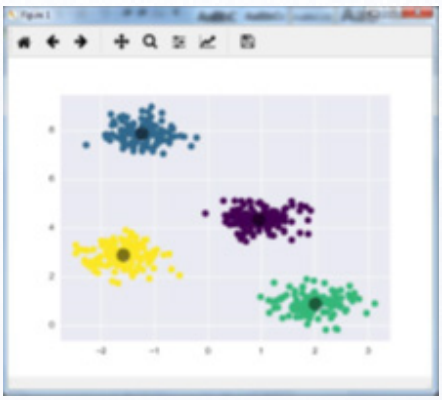

Figure 5: Visualization of data clustering

The entire Python clustering code is provided in the following.

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn.cluster import KMeans

from sklearn.datasets.samples_generator import make_blobs

# Generating data that was used for the example
# the make_blobs method generates a series of vectors that we will use to test
# the n_samples attribute represents the number of points to be generated
# n_features represents the number of characteristics each vector needs to have
# centers represent the number of groups, that is, the number of centers around which the vectors should be grouped
# cluster_std represents the standard deviation of point deviations from the center
# random_state number based on which generates
# X represents a variable in which a series of points should be placed
# y_true is a string that indicates which cluster belongs to which point
X, y_true = make_blobs(n_samples=1000, n_features=2, centers=4, cluster_std=0.40, random_state=0)

# the KMeans object is created to which the number of expected clusters is being forwarded as an argument
kmeans = KMeans(n_clusters=4)

# training KMeans model
kmeans.fit(X)

#by using the predict method of K-Means object the model prediction is made
y_kmeans = kmeans.predict(X)

# creating a scatter diagram based on the x and y coordinates of the generated data
# c attribute defines the color or group to which each element belongs
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

# the cluster_centers_ attribute represents the cluster center coordinates
centers = kmeans.cluster_centers_

# adds clusters to the existing diagram
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);

# displays the diagram on the screen
plt.show()
```

CONCLUSION

The main goal of the paper is to illustrate the capabilities of Python programming language implementation in AI applications. The main machine learning algorithms are realized using Python programming language, and the corresponding procedures and codes are demonstrated. The provided examples of Python programming language implementation can be a helpful guide for the research and development, government and industry communities as to how to apply Python in different contexts

related to AI field. Our future work will include a more detailed illustration of Python programming language implementation in the advanced AI-based applications.

ACKNOWLEDGEMENT

The authors would like to thank Ms. Jelena Misic for her support during this work and constructive suggestions. We would also like to thank the editor and anonymous reviewers for their comments which notably improve the paper content and quality.

REFERENCES

1. Nagpal A. and Gabrani G. (2019), "Python for Data Analytics, Scientific and Technical Applications," *Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates*, 2019, pp. 140-145, DOI: 10.1109/AICAI.2019.8701341.
2. James, G., Witten, D., Hastie, T., Tibshirani, R. (2013) "An Introduction to Statistical Learning", Springer New York Heidelberg Dordrecht London, ISBN: 978-1-4614-7137-0, DOI: 10.1007/978-1-4614-7138-7.
3. James, G., (2017), "Introduction to Statistical Learning – with Applications in R", Springer New York Heidelberg Dordrecht London, ISSN: 1431-875X, ISBN: 978-1-4614-7138-7 (eBook)
4. Hugunin. J., (1999) "Python and Java: The Best of Both Worlds", Proceedings of the 6th Annual Python Conference Available: http://www.python.org/workshops/1997-10/proceedings/hugunin.html
5. "NumPy documentation,", (2020), [Online] Available: https://docs.scipy.org/doc/NumPy/index.html
6. "Pandas documentation,", (2020), [Online] Available: https://Pandas.pydata.org/
7. "Python documentation,", (2020), [Online] Available: https://docs. Python.org/3/
8. Nanz, S., and Furia, A.C., (2015), "A Comparative Study of Programming Languages in Rosetta Code", Proceedings of the 37th International Conference on Software Engineering (ICSE'15), pages 778-788. IEEE, DOI: 10.1109/ICSE.2015.90
9. Shalev-Shwartzm, S., Ben-David, S., (2014), "Understanding Machine Learning: From Theory to Algorithms", Cambridge University Press., ISBN: 978-1-107-05713-5.
10. "SkitLearn documentation," (2020), [Online] Available: https://Scikitlearn.org/stable/index.html
11. Haslwanter, T.,(2016), "An Introduction to Statistics with Python with Applications in the Life Sciences", Springer, ISBN: 978-3319283159.