

Designing Generic Python based Micro Service for Machine Learning

Ashay Sant¹, S. R. Khone²

¹ME Second Year Student, Computer Engineering, Modern Education Society's College of Engineering, Pune, India

²Assistant Professor, Computer Engineering, Modern Education Society's College of Engineering, Pune, India

Abstract - Designing a generic machine learning service based on REST platform for prediction by using a combined tensorflow and scikit learn approach in python. The Data preprocessing being different for Google Tensorflow and scikit-learn, using factory pattern and delegator pattern, to generify this. The generic machine learning service combines use of Random Forest classifier, SVM classifier from scikit-learn and Google Tensorflow Deep Neural Network for class based classification. There is Abstract Machine Learning odelFactory that has responsibility of abstracting out the internal details that are implementation specific. This helps in expanding the code in future for multiple algorithms. Using Abstract Ensemble Learning Service, the microservice is enabled with ensemble voting capability. The code design lies on the formulation of Factory Pattern, Delegator Pattern, and Iterator pattern and is based upon Object Oriented Framework and is fully version controlled using private Stash repository. The feature vectors are represented using JSON and the microservice is designed in such a way that the feature vectors can be changed in the JSON schema to make the model feature vectors easily modifiable and scalable. To add whole new set of features, a new wrapper service and a data Model for particular purpose need to be added and the core Machine Learning functionality is generic and need not be altered.

Key Words: Design Patterns, MicroService, Delegator, Abstract class, Provider pattern, Tensorflow, scikit-learn, IntelliJ IDE.

1. INTRODUCTION

We aim to solve the problem of tight coupling of the Machine Learning Service to the problem at hand. We aim at making a horizontal ML service which would be robust and easily tunable from the environment variables. This will help to reuse the service for multiple purposes. The core logic of training and evaluating the model are very well abstracted out and are extensible as and when required.

When such a service is deployed, the models that are trained can be stored in cache or memory with the use of databases. This will solve the problem of reusing models again and again. The models can be retrained with particular frequent spanning a week or so. The deployment can be extended to production like environments as well.

The objectives are to design Machine Learning as a service. This can be done by formulating component diagram and class diagram. This will assist the developer in making the implementation easily. Also we need to formulate the existing assessments for the approaches already taken in this area. We need to check all the alternatives that could be used along with tensorflow and scikit-learn. We need to allow extensibility for future development as well.

In the said literature the authors have created a spread sheet based machine learning service. The data can be uploaded in the form of

1. JSON
2. xml
3. xlsx
4. csv

The user has the ability to upload multiple files at once and then these all files are combined using the attribute matching processing. The authors claim to handle the missing data as well. The missing data is handled using classification and regression analysis. The calculated data is filled in the same format as denoted by the input data format. The data types are matched. The uploaded files are transferred using servlets and then stored to Data Access Objects (DAO layer) and then stored in the database. The authors have used MLP machine learning algorithm. The authors also decide on the type of Machine learning from the datatypes. If the datatypes are numbers then regression analysis is performed. If the data values are repeated then the author proposes categorical data based classification like scheme. There is rematching because the MLP is regression instead of classification. The classification results are stored in the database. Further the servlet reads the results file from the database using DAO layer are transfers the results to the web application[1]. The accuracies achieved are

Data Shape	Blanks	Class distribution	Error Rate
600 tuples	40	10	21.35
200 tuples	40	10	14.24
600 tuples	80	20	10.13
400 tuples	40	10	9.2
600 tuples	120	30	11.2
600 tuples	120	10	19.35

The authors propose a microservice that is more reusable in the microservice clustered environment Three ML algorithms are focused which are

1. Feed-Forward
2. Neural Networks (FFNN), Deep Believe Networks (DBN), and
3. Recurrent Neural Networks (RNN).

The authors propose separation of actual implementation and abstraction of the internal details of ML service into microservice modules. The microservice is bifurcated into static and dynamic components. Instead of writing the boiler plate ML code, we can simply reuse the static ML components and adjust the dynamic ML components as per application needs. The author describes the VSL framework which stands for Virtual State Layer. It is a middleware that provides interface for

1. data exchanging for microservices
2. Service discovery
3. Late binding for data

VSL uses REST interface internally. The different protocols for VSL are

1. GET
2. SET
3. SUBSCRIBE and
4. NOTIFY

The authors explain the sample xml configuration written in XML format. The xml schema is like

```

<MLConf42 type="FFNNConfig">
<InputDataAddresses>
</InputDataAddresses>

<OutputDataAddresses>
</OutputDataAddresses>
</MLConf42>

```

The <InputDataAddresses> have the address for the source data node. This means that the data that is input is present in <InputDataAddresses> nodes. The <OutputDataAddresses> represents the node address to which the output will be stored from Machine Learning the input data. The authors have used internally Google's tensorflow library in python. The authors have used MNIST dataset for testing their model.

1. Time comparison -

The author compares direct implementation to that of Machine Learning as a service approach. The time difference for different algorithms is
Algorithm plain MLaaS

FFNN 0.26s 0.39s

DBN 2s 4s

RNN 2s 10s

The differences are due to the latency of passing configuration.

2. Performance

There is some overhead for making the internal calls to google tensorflow.

3. Cognitive Assessment

Algorithm	Task	Bare library	Bare Lib LOC (lines of code)	Author's approach	LOC
FFNN	impl	5 min	85	30s	2
FFNN	setup	2 min		30s	
DBN	impl	8 min	148	30s	2
DBN	setup	3 min		30s	
RNN	impl	10min	128	30s	2
RNN	setup	5 min		30s	

The MLaaS reduces the development time. In the above table the authors compare the time required to change the dynamic parameters of the service and time required for setup using raw tensorflow API's [2]

The authors have solved the problem of finding a suitable configuration for an electric motor with the help of Azure Machine Learning studio. The various parameters of induction motor which are to be optimized include

1. Power
2. Voltage
3. Speed
4. Frequency
5. Strator Core Length (Inductor)
6. Strator Core inner Diameter
7. Efficiency

A Java program was written using NetBeands IDE to simulate working of an induction motor. The combination of Strator Core Length vs Strator Core inner Diameter was optimized using exhaustive approach. This is the case for acquiring highest efficiency. The Azure ML studio was used which offers a very friendly experience with a flowchart like implementation. All the internal details are abstracted from the user. ML in azure learning studio was made to work by selecting an intended minimization/maximization function like

1. maximizing accuracy
2. minimizing root mean square error
3. increasing precision and recall

All the combinations of the input parameters are enumerated and for each combination a metric is selected.

After evaluating using ML algorithms, we get the combination for which the output function is Maximum (Minimum). We have 2 kinds of experiments in Azure ML Studio

1. Training experiment
2. Predictive experiment

The model can also be published as a web service wherein REST based calls can be used. The authors made a choice of Multiclass Neural Network and Boosted Decision Tree Regression.

The authors made a conclusion wherein the time for predicting all the parameters for highest induction motor efficiency as 1 min 45s compared to traditional enumeration which takes 15 mins [3].

The authors have used Amazon S3 storage (Amazon Simple Storage Service) to store the dataset which they acquired from "University of California Irvine Machine Learning Repository". The data is preprocessed before storing it to the S3 storage. Yes and No attribute values are stored as numbers represented by 1 and 0. After that a data source is created by logging into AWS ML console. The three ML models supported by Amazon are

1. Binary classification model
2. Multiclass classification model
3. regression model

The authors have described about Binary classification. Amazon has made use of Logistic Regression for the purpose of Binary classification [4]. The author has explained deep neural network model by using tensorflow and MNIST (study dataset). The author has explained various activation functions TanH, Exponential Linear Unit (eLu), Softplus, Softsign, Rectified Linear Unit (ReLU). Per the author most accurate results are obtained using ReLU for MNIST dataset. The author has demonstrated the accuracy of each and every activation function. The TanH method was the least accurate with accuracy of 0.75 and highest was ReLU with 0.98. The proposed System has the following phases

1. Preparing the MNIST dataset
2. Determination of Feature Vectors
3. Selection of Weights for DNN
4. Selection of activation functions
5. Getting results

In the MNIST dataset we have a total of 60,000 figures for training and 10,000 figures for prediction as test dataset [12].

The author has created a machine learning model for fault classification in Power Systems. The power system converts one form of energy to another and uses gears, motors and shafts to transfer power from one component to another. The dataset is simulated in Matlab for 100KM overhead power lines and there are total of 11 faults i.e. class labels. Simulink is an add-on for Matlab which allows simulation based on randomness. Scikit-learn was used to train supervised learning models in python. There were total of 3 algorithms tested which are

- a. SVM
- b. KNN
- c. Decision Tree

The author describes differences among supervised and unsupervised machine learning approaches. The Feature Vectors include 3 set of RMS voltages and 3 set of RMS current values. The output is the type of fault. Categorical data for class labels is converted into encoded numerical values using sklearn. Preprocessing. The algorithms which gave the least accuracy was Decision Tree with accuracy of 84.2%. The SVM gave the highest accuracy of 91.6%. The kNN stands in between these 2 with accuracy of 86.15%. The total training tuples included 80% of 11300 tuples. The remaining were used as test dataset. The predicted label is compared with actual class label of the test dataset [17]. Various free libraries along with categorization is described by the author. The segregation done by author are based on

- a. Data Preparing
- b. Core Libraries
- c. Data Visualization
- d. Machine Learning
- e. Deep Learning
- f. Big Data

The Core Libraries explained include SciPy, NumPy and NetLib. Cython is an additional benefit of using python as it facilitates calling C functions directly. The reputation on GitHub about various libraries is also shown. Numpy and pandas are highly scored here. Then we turn our focus on Data preparation libraries wherein we have pandas. For Data Visualization we have Matplotlib, seaborn and plotly. For machine learning we have tensorflow for Deep Neural Networks and scikit-learn. Also keras is a machine learning library made on top of TensorFlow. PyTorch is more recent one developed by facebook and with a good community support. [18]. Video interviews are being analyzed for enhanced Human Computer Interaction and also psychological assessment. CNN models can identify such traits and tag their personality. 120 real job applicants were studied with AI interviewing system and personal traits and expressions were detected using TensorFlow AI engine. The big five traits of humans are

- a. Openness
- b. Conscientiousness or Thoughtfulness
- c. Extraversion
- d. Agreeableness
- e. Neuroticism (Reflects anxiety)

One AVI (Asynchronous Video Interview) server was created for data collection using google cloud. Data labelling was done via a survey which every participant needed to take before opting for the conference. Features were extracted from 86 landmark points. Before that videos were converted

to images and then the resizing of the images was done. The prediction was carried out by the model created using Python and TensorFlow DL engine. Each feature was normalized on a scale of [0,1] before feeding it to the predictor. The accuracy of this experiment was 95.36% [19]

2. DESIGNING THE MICRO-SERVICE

Firstly we need to design a template so that the boiler plate code which we write is extensible and easy to maintain. For this purpose we made use of the Object Oriented Design Patterns.

The primary design which comes into picture here is the Abstract Service Pattern. We have Abstract Machine Learning ModelService which defines specific structure and the core implementation details can be abstracted out. The AbstractMLModelService defines all the methods that are necessary for any model to function. The methods defined are –

1. model_accuracy(self)
2. create_model(self, feature_data, label_data)
3. predict(self, feature_data)
4. evaluate_model(self, trained_model, feature_data, label_data)

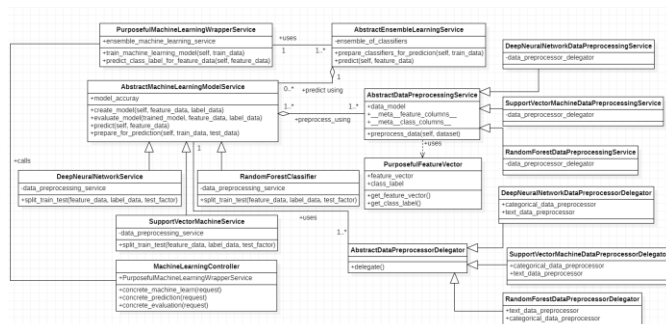


Fig -1 Class Diagram For Micro-Service

The other component of the Micro-Service are

1. MachineLearningController
2. AbstractMachineLearningModelService
3. PurposefulMachineLearningWrapperService
4. AbstractEnsembleLearningService
5. PurposefulFeatureVector
6. AbstractDataPreprocessor
7. AbstractDataPreprocessorDelegator

2.1 The Workflow

The rest request is sent by the user which reaches the MachineLearningServiceController. In the controller, the input request can be deserialized using serializers. After that the request is handed over to the respective PurposefulMLServiceWrapper. From here using PurposefulMachineLearningWrapperService the different ML models can be trained.

When training the model we require the delegator to preprocess Text and Categorical data. AbstractDataPreprocessorDelegator delegates the data to the respective preprocessor which extends AbstractDataPreprocessor.

The PurposefulFeatureVector is linked to PurposefulMachineLearningWrapperService. From here the model which is to be learnt picks up the feature vectors.

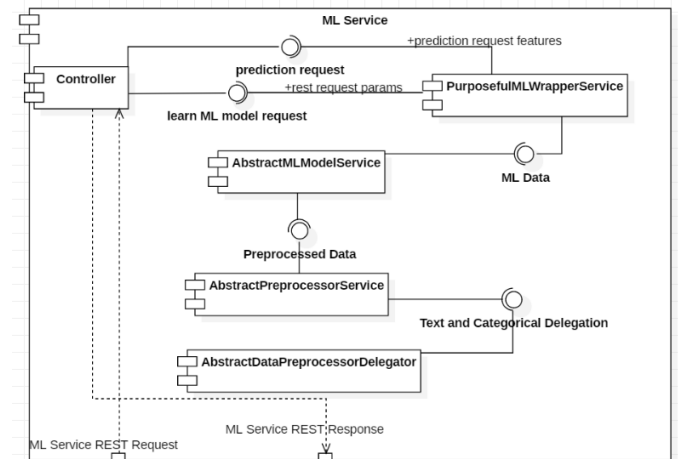


Fig -2 Workflow Representation as Component Diagram

3. THE DESIGN PATTERNS FOR EXTENSIBILITY

To design an extensible micro-service it is essential to use design patterns so that it follows the principle of “convention over configuration”. This principle states that there is a proper structure that we need to follow so that the code is maintainable. By this principle we have enclosed every object creation functionality in Container classes. The design patterns that are used include –

1. Delegator Pattern
2. Iterator Pattern
3. Provider Pattern (Analogous to Spring Beans)
4. Chain of Responsibility
5. Inversion of Control

3.1 Delegator Pattern

The respective preprocessor Delegator has the task of delegating the preprocessing to the preprocessor. The delegator checks the datatype of incoming feature vector and then calls the respective preprocessor.

3.2 Iterator Pattern

This pattern is used to iterate through the difference feature vectors and if ensemble is used to iterate via various implementations. The iterator is just a simple for loop or a complex iterator object for different types of lists.

3.3 Provider Pattern

This pattern is used to manage the dependencies. Example of the Provider pattern for dependency injection in python is given below –

```
import collections
import dependency_injector.providers as providers

deep_neural_network_text_data_preprocessor_factory =
providers.Factory(DeepNeuralNetworkTextDataPreprocessor)
deep_neural_network_text_data_preprocessor_01 =
deep_neural_network_text_data_preprocessor_factory()
deep_neural_network_text_data_preprocessor_02 =
deep_neural_network_text_data_preprocessor_factory()
```

3.4 Chain of Responsibility

When we are using PurposefulService for representing a concrete problem, we can have multiple instances of such PurposefulService. We need to pass in distinguishing info in the rest request itself. Then the controller can check which PurposefulHandler is suitable. This forms chain of multiple PurposefulService. The selection can be accomplished with the help of a supports (credentials) method.

3.5 Inversion of Control

The responsibility of the Object creation is up to the provider pattern in the python. The dependencies are injected by using Constructor Injection. The methods of dependency injection are -

1. Constructor Injection
2. Setter Injection

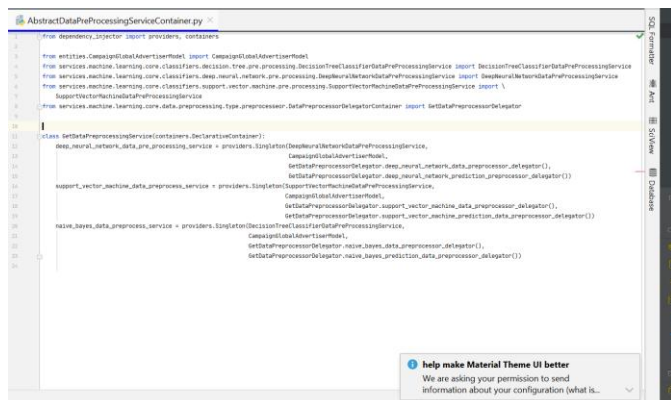


Fig -3: Dependency Injection Using Provider Pattern

4. EXCEL EVALUATION SERVICE

The excel evaluation service is used for comparing the results of individual predictions. The purpose of this component is to write the accuracy metrics to an excel sheet.

From this excel sheet we can apply transformations, calculate mean and generate plots. This plays major role in data visualization. These graphs can be programmatically generated in the excel sheet as well. The main methods provided by this service are –

1. write headers(sheet name)
2. get next row num(sheet name)
3. append to sheet by column(sheet name, row, col, value)
4. append dictionary data to sheet(sheet name, data dict)
- 5.

The service is static in nature and need not be instantiated in order to user its methods. This is more of utility service that is an add on to the base Machine Learning Service.

The excel output of the service consists of -

1. Feature Vector Data Shape
2. Train Test Split Factor
3. Model Name
4. Model Accuracy
5. Model specific parameters like kernel function, batch size, number of epochs

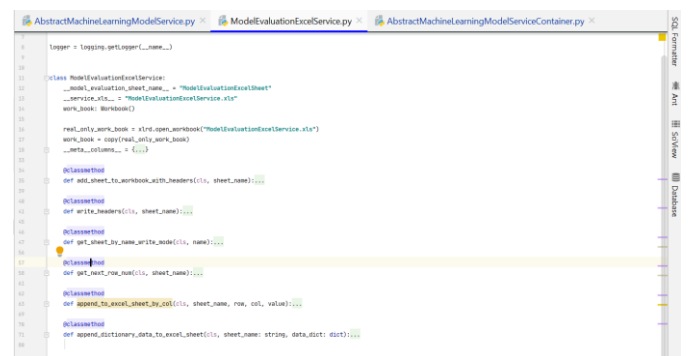


Fig -4: ModelEvaluationExcelService

Train	Feat	Train_Lab	Train_Lab_Split	Model_Nar	Accuracy	Model_Clr	DeepNeur	DeepNeur	DeepNeur	SupportVe	SupportVector	Machine_Regularization
25385	23	25385	1	SupportVe	0.741581	284.8421						
21154	23	21154	1	0.25 SupportVe	0.727028	195.4191				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	224.38				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	193.4382				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	184.7506				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	186.3176				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	190.3829				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	197.0841				rbf	1	
21154	23	21154	1	0.25 SupportVe	0.727028	215.3829				rbf	1	
21154	23	21154	22	0.25 DeepNeur	0.816222	12.7393	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.836642	11.69116	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.816931	12.78789	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.809841	15.0842	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.757523	14.15977	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.802893	10.47141	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.843023	13.30318	8	32	TRUE			
21154	23	21154	22	0.25 DeepNeur	0.819931	10.99141	8	32	TRUE			
25385	23	25385	22	DeepNeur	0.847572	13.16713						
21154	23	21154	1	DecisionTr	0.999575	1.930648						
25385	23	25385	1	SupportVe	0.741581	414.7354						
25385	23	25385	22	DeepNeur	0.846154	12.99193						
			1004	7	Ensemble	0.779291						
			1024	7	Ensemble	0.771484						

Fig -5: Excel Service generated Tables

5. ADVANTAGES AND LIMITATIONS

The Feature Vector can be changed based on the purpose. The features and class label are picked from JSON schema. The PurposefulService, its DataEntity and JSON feature factors need to be added along with a controller mapping so

as to use the generic service. AbstractModelService makes it easy and perform implementation independent operations.

Hybrid approach of scikit-learn, tensorflow and in future pytorch also could be added. REST based interface so that communication is independent of implementation. The service is designed using Django framework and is totally docker friendly. AbstractServices are created wherever possible so that code is easy to extend and maintain. The service is version controlled using git. Private or public repository can be used to maintain the code. This enables multiple developers to add new features.

All the dependencies are freezed using pip in requirements.txt. When setting up the project for the first time, dependencies can be picked up from requirements.txt file

Since we are enforcing on using design patterns, the main focus is on the code structure. To maintain such structure more of code is required. The Generic ML service is based on Object Oriented framework. The key terms like inheritance, Abstract Base class, providers are must for working on this service.

As we are using REST framework for the service based communication there is additional overhead of marshalling request and response DTO's. The Django framework brings a lot of transitive dependencies and network connectivity is required to get these dependencies. These are bare minimum for starting the service locally/ on the server instance. Whenever we issue a query to the database there is implicit translation to Django entities since we use Entity Management framework. This may take too long for complex queries and here the native SQL's can help reduce the query time using proper optimization techniques. The determination of the hyper parameters for the ML service are NP type problem. Hence it cannot be guaranteed to be accurate 100%. The training time for model gets long when several models are trained repetitively for evaluation.

5. CONCLUSIONS AND FUTURE SCOPE

5.1 Framework and Communication

We can deploy machine learning as a service. The service will be based on Django framework and python as the programming language. We solve the problem of inter-service communication by using REST framework and intra-service communication using Object Oriented framework.

5.2 Dynamic Feature Vectors

We also can provide a JSON schema to hold the Feature Vectors for training the model. The purposeful wrapper service plays an important part as it interacts with the core Entities (Repository). For solving a new problem we just

need to define the concrete Data Entities and concrete Purposeful Service.

5.3 Future Scope

The service designed can be containerized and deployed onto docket and kubernetes. This will enable it to make to various non-production and production environments easily. Also we can override the application level properties to instruct a service to run the model evaluation tasks. The service can be extended to include pytorch framework alongside tensor ow and scikit-learn. This will enable more robust behavior. A cross framework ensemble can also be trained.

REFERENCES

- [1] U. Spreadsheets, C. Yoon and S. Kang, "A study on machine learning web service," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 760-765
- [2] M. Pahl and M. Loipfinger, "Machine learning as a reusable microservice," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, 2018, pp. 1-7
- [3] V. Pliuhin, M. Pan, V. Yesina and M. Sukhonos, "Using Azure Maching Learning Cloud Technology for Electric Machines Optimization," 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology, Kharkiv, Ukraine, 2018, pp. 55-58
- [4] R. Ramesh, "Predictive analytics for banking user data using AWS Machine Learning cloud service," 2017 2nd International Conference on Computing and Communications Technologies (ICCT), Chennai, 2017, pp. 210-215.
- [5] Dalibor Bui, Jasminka Doba, \Lyrics classification using Naive Bayes", 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, Opatija, Croatia, 02 July 2018, PP 1011-1015
- [6] O. Obulesu, M. Mahendra, M. ThrilokReddy, \Machine Learning Techniques and Tools: A Survey", 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, Coimbatore, India, 03 January 2019, PP 605-611
- [7] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, Zhen Liu, \An Adaptive En-semble Machine Learning Model for Intrusion Detection", IEEE Access(Volume 7), IEEE, 19 June 2019, PP 82512-82521
- [8] Xiaobo Liu, Zhentao Liu, Guangjun Wang, Zhihua Cai, Harry Zhang, \Ensemble Transfer Learning Algorithm", IEEE Access(Volume 6), IEEE, 13 December 2017, PP 2389-2396

- [9] Anjan Kumar Amirishetty, Yunrui Li, Tolga Yurek, Mahesh Girkar, Wilson Chan, Graham Ivey, Vsevolod Panteleenko, Ken Wong, "Improving Predictable Shared-Disk Clusters Performance for Database Clouds", 2017 IEEE 33rd International Conference on Data Engineering (ICDE), IEEE, San Diego, CA, USA, 18 May 2017, PP 237 -242
- [10] White Paper from Oracle, "Oracle NoSQL Database for Time Series Data", December 2017
- [11] Chetan Jaiswal, Vijay Kumar, "DBaaS: Database High Availability as a Service", 2015 11th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), IEEE, Bangkok, Thailand, 08 February 2016, PP 725-732
- [12] Vasileios Tsoukas, Konstantinos Kolomvatsos, Vasileios Chioktour, Athanasios Kakaroun-tas, "A Comparative Assessment of Machine Learning Algorithms for Events Detection", 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), IEEE, Piraeus, Greece, 21 November 2019, PP 1-4
- [13] AJAY SHRESTHA AND AUSIF MAHMOOD, "Review of Deep Learning Algorithm-based Architectures", IEEE Access (Volume 7), IEEE, 22 April 2019, PP 53040-53065
- [14] Stelios E. Papadakis, Vangelis A. Stykas, George Mastorakis¹ and Constandinos X. Mavromoustakis, "A hyper-box approach using relational databases for large scale machine learning", 2014 International Conference on Telecommunications and Multi-media (TEMU), IEEE, Heraklion, Greece, 09 October 2014, PP 69-73
- [15] Hanane Bais, Mustapha Machkour, Lahcen Koutti, "Querying Database using a universal Natural Language Interface Based on Machine Learning", 2016 International Conference on Information Technology for Organizations Development (IT4OD), IEEE, Fez, Morocco, 26 May 2016, PP 1-6
- [16] Fatih Ertam, Galip Aydin, "Data Classification with Deep Learning using TensorFlow", 2017 International Conference on Computer Science and Engineering (UBMK), IEEE, Antalya, Turkey, 02 November 2017, PP 755 - 758
- [17] T. Goswami and U. B. Roy, "Predictive Model for Classification of Power System Faults using Machine Learning," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 1881-1885
- [18] I. Stanin and A. Jovi, "An overview and comparison of free Python libraries for data mining and big data analysis," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019, pp. 977-982
- [19] H. Suen, K. Hung and C. Lin, "TensorFlow-Based Automatic Personality Recognition Used in Asynchronous Video Interviews," in IEEE Access, vol. 7, pp. 61018-61023, 2019
- [20] N. Ari and M. Ustazhanov, "Matplotlib in python," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, 2014, pp. 1-6
- [21] D. Prusti and S. K. Rath, "Fraudulent Transaction Detection in Credit Card by Applying Ensemble Machine Learning techniques," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-6