

Adappt Tasks - Machine Learning model documentations

Project: To predict the attrition risks (exited) for the employees

- The process of predicting employee attrition risks entails creating a model that calculates the probability of a worker leaving the company. This aids companies in efficiently managing their employees.
- In order to reduce the risk of attrition, the process involves gathering employee data, preprocessing it, choosing pertinent features, training a predictive model using machine learning algorithms, assessing the model's performance, deploying it for predictions, and taking the necessary actions based on the predictions.
- To guarantee the model's accuracy over time, ongoing monitoring and upgrades are required.

List of Parameters:

- corporation
- lastmonth_activity
- lastyear_activity
- number_of_employees
- exited

Steps followed while implementing the model:

- Loaded two input datasets (dataset1.csv and dataset2.csv) and combined it to one dataset (combined.csv)
- Describing the dataframes using describe function
- Checking for Null values. There were no null values.
- Checked for duplicate values. There were 10. Removed the duplicate values
- Printed the Unique values from the data frame.
- Performed Data Visualisation by plotting Histogram, Colorbar, Heatmap, Barograph, Distplot, Countplot, Scatter plot, etc.
- Number of employees are more than 100 in the corporation 'bqlx'
- Exited employees are more than current employees
- In the last month activity the current employees are more than exited employees
- Dropping 'exited' column for further modeling.
- Defining X and Y variables. Training a logistic regression model on the input dataset, predict the target variable for the test dataset, and print the coefficients of the trained model.
- Using sklearn-metrics the values of f1 score, Recall, Precision and Accuracy can be seen. F1-score: 85%, Recall: 87%, Precision: 89% and Accuracy: 87%.
- Linear Support Vector Classifier was used to test further accuracy by setting the learning rate and regularisation parameters. The accuracy obtained was 75%
- Trained a decision tree classifier model using scikit-learn's DecisionTreeClassifier class with a specific maximum depth and maximum leaf nodes (2 and 7). The training score obtained was 90% and testing score was 87%

- Trained a Random Forest classifier model on this dataset using scikit-learn's RandomForestClassifier class with a specific maximum depth and maximum leaf nodes. (2 and 7). The training score obtained was 88% and testing score was 87%.

Note: The accuracies obtained was 87%. It can be varied when tuning of hyper parameters are done. Kindly re-run for better results.

Steps followed while deploying the model:

1. To run the ML model: Import all the necessary packages

Use: python3 employee.py

2. To run the Docker through CLI:

a. Open Terminal. Change the directory where it contains the model and datasets.

b. Create a docker file, by entering the below commands. Copy paste it and modify the directory

```
# Use a base image with the desired operating system and runtime
FROM python:3.9-slim-buster
```

```
# Set the working directory in the container
WORKDIR /Users/sudharmendragv/Downloads/Adappt
```

```
# Copy the ML model files to the working directory
#COPY /Users/sudharmendragv/Downloads/Adappt .
COPY requirements.txt .
```

```
# Install the required dependencies
RUN pip install --no-cache-dir -r requirements.txt
```

```
COPY . .
```

```
# Expose any necessary ports (if applicable)
#EXPOSE 8000
```

```
# Define the command to run your ML model
CMD ["python", "employee.py"]
```

c. Save the dockerfile in the same directory and create a requirement file with all the necessary packages details. This should look like below.
requirements.txt

```
numpy==1.19.5
pandas==1.3.4
scikit-learn==1.0
matplotlib==3.4.3
seaborn==0.11.2
scipy==1.7.1
```

d. Save the requirements.txt file and run the below command.

3. Build the docker image: "docker build -t ml_model ."

Note: "ml_model" - Name of the docker image. Any name can be provided

Run the docker image: docker run ml_model

Optimisations:

Implemented couple of models: Logistic Regression, Linear SVM, Decision Tree and Random Forest. Obtained an accuracy of 87%

Inorder to obtain the same accuracy, hypertuning of the parameters were done like:

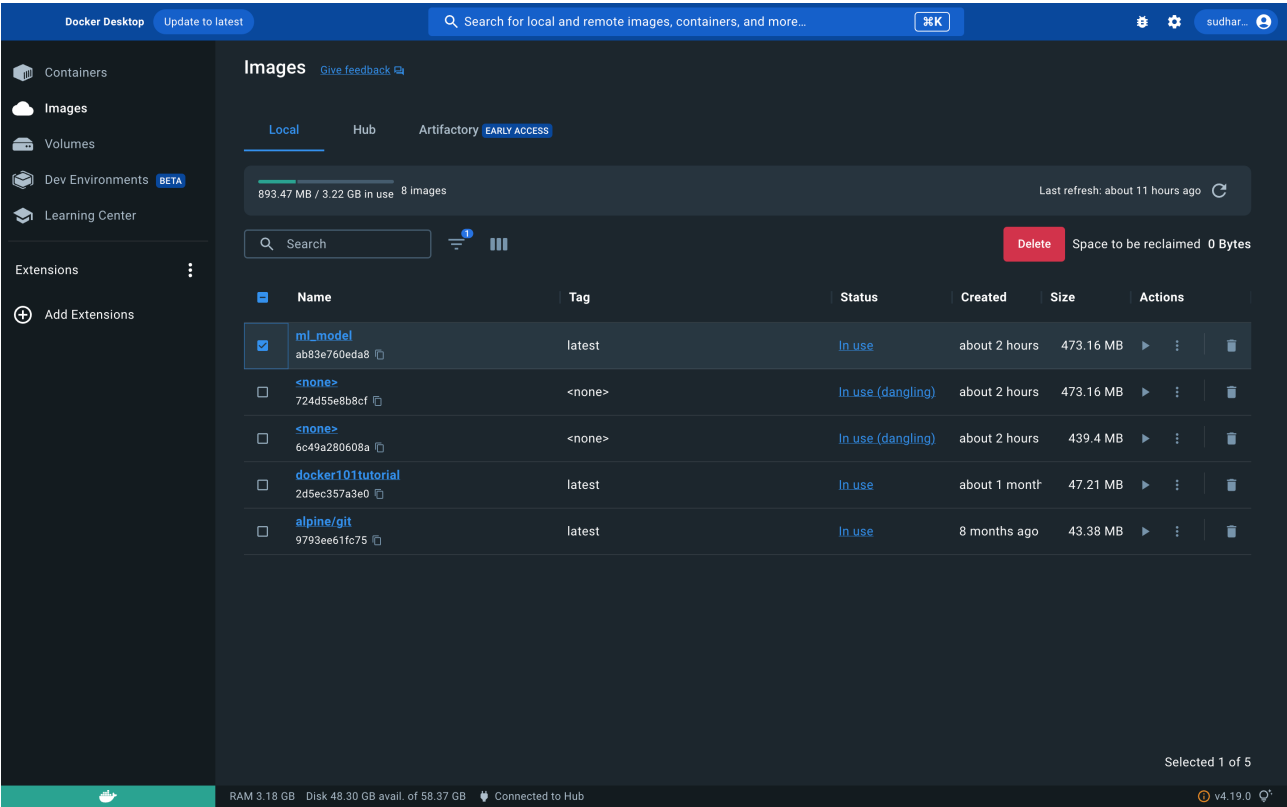
- For Logistic Regression: Tuning the RandomState value
- For Linear SVC: Tuning the learning rate and regularization parameters

Output screenshots:

1. CLI Output:

```
(base) sudharmendragv@Sudharmendras-MacBook-Air Adappt % docker run ml_model
corporation:26
lastmonth_activity:24
lastyear_activity:26
number_of_employees:22
exited:2
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   corporation            36 non-null     object
1   lastmonth_activity     36 non-null     int64
2   lastyear_activity      36 non-null     int64
3   number_of_employees    36 non-null     int64
4   exited                36 non-null     int64
dtypes: int64(4), object(1)
memory usage: 1.5+ KB
Number of duplicates: 0
   lastmonth_activity  lastyear_activity  number_of_employees  exited
0                100                1359                 1         0
1                 68                 282                 14         0
2                 71                 949                  40         1
3                684                3782                 103         0
4                 45                 655                  7         0
5                  0                  18                  21         1
6                189                 961                  18         1
7                 16                1823                  33         0
8                  9                  45                   1         1
9                  0                  67                   14         1
10                 48                 986                  22         1
11                 52                 650                   11         1
12                1090                2452                   9         0
13                  6                  88                   90         1
14                 99                 390                   99         1
15                 75                 800                   81         1
16                285                1687                   2         0
17                 78                1024                   12         1
18                 14                2145                   20         0
19                182                3891                   35         0
20                181                10963                   2         1
21                  0                  113                   42         1
22                929                1992                   1         0
23                 19                 455                   8         1
24                 94                 868                   3         1
25                 81                1401                   10         0
(26, 3)
(26,)
Printing the co-efficients : [[-0.00709661  0.00018846  0.02508652]]
F1-score: 0.8589743589743589
Recall: 0.875
Precision: 0.8928571428571428
Accuracy: 0.875
Confusion-Matrix :
[[1 1]
 [0 6]]
Train score: 0.5
Test score: 0.75
Train score: 0.9444444444444444
Test score: 0.875
Train score: 0.8888888888888888
Test score: 0.875
(base) sudharmendragv@Sudharmendras-MacBook-Air Adappt %
```

2. Docker Image:



3. Docker Output:

