Robot Classify

Release Alpha

Scott R Smith

Feb 15, 2020

CONTENTS

CHAPTER

ONE

INSTALATION AND OVERVIEW

1.1 Introduction

RobotClassify allows for non-data scientest such as citizen developers and other operational people involved with analizing and reporting on business data. The goal is to automate the entire ML process (feature-engineering, training, prediction).

This version of the app is optimized for loading datafiles to train with, and test files for predctiona, optimzed for submission in Kaggle comoetitions

My motivation for project centers around my interest in machine learning for citizen developers. Taking the compleated tasked of feature engineering, model selection, and training and makeing it a simple point anc click excersie without any prior ML Training.

1.2 Project dependencies, local development and hosting instructions

- Detailed instructions for scripts to install any project dependencies, and to run the development server.
- Documentation of API behavior and RBAC controls

1.3 Runing and Testing instrunctions

URL: Auth: Testing: https://robotclassify.herokuapp.com/

1.4 Getting Started

1.4.1 Installing Dependencies

1.4.2 Python 3.7

This project uses python 3.7

To Install Python

1.4.3 PIP Dependencies

Once you have your virtual environment setup and running, install dependencies by navigating to the root directory and running:

```
pip install -r requirements.txt
```

This will install all of the required packages we selected within the `requirements.txt` file.

1.4.4 Key Dependencies

- Flask is a lightweight backend microservices framework.
- SQLAlchemy is the Python SQL toolkit and ORM.
- Flask-CORS is the extension used to handle cross-origin requests from the frontend server.
- Auth0 Provides authentication and authorization as a service
- Postgres DOES XXX
- Heroku DOES XXX
- Flask-WTF DOES XXX
- mlLib DOES XXX
- InitTest DOES XXX
- FlaskMigrate DOES XXX

1.5 Database Setup

The app is running Postgres SQL.

1.6 Running the server

From within the `root` directory to run the server, execute:

```
export FLASK_APP=app.py
export FLASK_ENV=development
flask run
```

1.7 Documentation

1.7.1 Opening the API Documentation

Documentation is generated with Sphinx.

HTML Documentation

```
From the root folder, open the index file in a browser

.. code-block:: bash

   ./docs/build/html/index.html

PDF Documentation
~~~~~~~~~~~~~~~~~

The PDF version of the documentation is located in the root project directory. Named␣
↪RobotClassifyapi.pdf

Generating documentation
^^^^^^^^^^^^^^^^^^^^^^^^

Documentation is generated with Sphinx.

Installing Sphinx and support tools
~~~~~~~~~~~~~~~~~~
```

To install Sphinx, reference the documents at https://www.sphinx-doc.org/en/master/usage/installation.html

For example:

```
pip install -U sphinx
```

Install dependencies by navigating to the `root` project directory and running:

```
cd docs
pip install m2r
pip install recommonmark
pip install rinohtype
pip install -r requirements.txt
```

Generating the documentation ~~~

Generate the documentation with the following commands

```
# From the root project directory
# Convert readme to rst to be included in generated docs
m2r README.md README.rst --overwrite
cp -R README.rst ./docs/source
cd ./docs
make html
# Make pdf
make latexpdf
cd ..
cp -R ./docs/build/latex/RobotClassifyaapi.pdf .
```

1.8 API End Points

The following APIs are available. Detailed html documentation can be found in the 'docs/source' folder.

– Home Page –

- GET / (home)

– Documentation Page –

- GET /docs/index.html

— Projects —

- GET /projects (List all projects) - get:project
- GET /projects/int:project_id (Project page) - get:project
- POST/GET /projects/create (create a new project) - post:project
- PATCH /projects/int:project_id/edit (edit a project) - patch:project
- DELETE /projects//delete (Delete a project) - delete:project

— Runs —

- GET /runs/int:run_id (Display a run results) - get:run
- GET/POST /runs/create/int:project_id (Create a run) - get:post
- DELETE /runs/int:run_id/delete (Delete a run) - delete:post
- PATCH /run/int:run_id/edit (edit a run) - patch:run

— Train —

- GET /train/int:run_id (run ML training for a run) post:train
- GET /train/int:run_id/download (download testing results file, .. code-block:

  kaggle file) get:train

1.9 Error Handling

Errors are returned as JSON objects in the following format:

```
{
    "success": False,
    "error": 400,
    "message": "Bad Request"
}
```

The API returns multiple error types when requests fail:

- 400: Bad Request
- 404: Resource Not Found
- 405: Method Not Allowed
- 422: Not Processable
- 500: Internal Server Error

1.10 Testing

Testing is done with UnitTest. Load and run the test collection:

Auth0 Management API (Test Application)

1.11 Development Notes

- Flask Sessions are maintained between REST Calls for Web-based use of the API. The implementation is based upon
- CSRF protection is disabled for certain REST calls to faciliate testing cia CuRL.
- Patch and Delete functions are only avialable via API calls
- UnitTest uses a local postgres database
- UnitTest uses API App Auth0 credentials (verses using Auth0 Web App quickstart code) Auth0 Management API (Test Application)
- Tokens in the headers are used for API authentication

CHAPTER

TWO

ROBOT CLASSIFY'S API CONTROLLERS

Introduction

Home Page

- GET / (home)

Documentation Page

- GET /docs/index.html

Projects

- GET /projects (List all projects) - get:project
- GET /projects/<int:project_id> (Project page) - get:project
- POST/GET /projects/create (create a new project) - post:project
- PATCH /projects/<int:project_id>/edit (edit a project) - patch:project
- DELETE /projects/<project_id>/delete (Delete a project) - delete:project

Runs

- GET /runs/<int:run_id> (Display a run results) - get:run
- GET/POST /runs/create/<int:project_id> (Create a run) - get:post
- DELETE /runs/<int:run_id>/delete (Delete a run) - delete:post
- PATCH /run/<int:run_id>/edit (edit a run) - patch:run

Train

- GET /train/<int:run_id> (run ML training for a run) post:train
- **GET /train/<int:run_id>/download (download testing results file,** kaggle file) get:train

app.**get_token_from_auth0**()
Obtains the Access Token from the oAuth object

app.**get_token_from_header**()
Obtains the Access Token from the Authorization Header

app.**index**()
Home Page

Display the home page

- **Sample Call::** curl http://localhost:5000/
- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found: The requested URL..."
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**send_documents**(*path*)

Documentation Page

Display the documetnation pages

- Sample Call:

```
curl -X GET http://localhost:5000/docs/index.html
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found: The requested URL..."
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**projects**(*payload*)

List Projects

Display a list of projects

- Sample Call:

```
export TOKEN=...
curl -X GET http://localhost:5000/projects
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 302
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
```

- Expected Fail Response:

```
HTTP Status Code: 405
{
 "description": "405 Method Not Allowed...",
 "error": 405,
 "message": "Method Not Allowed",
 "success": false
}
```

app.**show_project**(*payload*, *project_id*)

Project

Display a single projects

- **Sample Call::**

  curl -X GET http://localhost:5000/projects/1 -H "Authorization: Bearer $TOKEN"

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
 HTTP Status Code: 404
{
 "description": "404 Not Found: The requested URL..."
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**create_projects_submission**(*payload*)

Create Project

Create Project

- Sample Call:

```
export TOKEN="edfgdfgd..."
curl -X POST http://localhost:5000/projects/create
     -H "Authorization: Bearer $TOKEN"
     -F "form-project-name=New Test Project"
     -F "form-project-description=Testing Project Create"
     -F "form-project-trainingFile=@examples/titanic_train.csv"
     -F "form-project-testingFile=@examples/titanic_test.csv"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
    "description": "401: Authorization header is expected.",
    "error": 401,
```

(continues on next page)

(continued from previous page)

```
    "message": "Unauthorized",
    "success": false
}
```

app.**edit_project_submission**(*payload*, *project_id*)

Edit Project

Edit Project

- Sample Call to edit:

```
curl -X PATCH http://localhost:5000/projects/1/edit
     -H "Authorization: Bearer $TOKEN"
     -F "form-project-name=Titanic Disaster Patch"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
    "description": "401: Authorization header is expected.",
    "error": 401,
    "message": "Unauthorized",
    "success": false
}
```

app.**search_projects**(*payload*)

Search Projects

Search Project

- Sample Call to search:

```
curl -X POST http://localhost:5000/projects/search
     -H "Authorization: Bearer $TOKEN"
     -F "search_term=Titanic"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
    "description": "401: Authorization header is expected.",
    "error": 401,
    "message": "Unauthorized",
    "success": false
}
```

app.**delete_project**(*payload*, *project_id*)

Delete Project

Delete Project

- Sample Call:

```
curl -X DELETE http://localhost:5000/projects/3/delete
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
HTTP Status Code: 200
{"success"}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found:  If you entered....",
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**show_run**(*payload*, *run_id*)

Runs

Display a single run

- Sample Call:

```
curl -X GET http://localhost:5000/runs/1
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found:  If you entered....",
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**create_run_submission**(*payload*, *project_id*)

Create Run

Create Run

- Sample Call:

```
curl -X POST http://localhost:5000/runs/create/1
     -H "Authorization: Bearer $TOKEN"
     -F "form-run-name=New Curl Run"
     -F "form-run-description=Via curl"
     -F "form-run-targetVariable=Survived"
```

(continues on next page)

(continued from previous page)

```
     -F "form-run-key=PassengerId"
     -F "form-run-predictSetOut=PassengerId"
     -F "form-run-predictSetOut=Survived"
     -F "form-run-scoring=f1"
     -F "form-run-modelList=xgbc"
     -F "form-run-basicAutoMethod=True"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
    "description": "401: Authorization header is expected.",
    "error": 401,
    "message": "Unauthorized",
    "success": false
}
```

app.**delete_run**(*payload*, *run_id*)

Delete Run

Delete Run

- Sample Call:

```
curl -X DELETE http://localhost:5000/runs//delete
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
 HTTP Status Code: 200
{'success'}
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
 "description": "404 Not Found: The requested URL was....",
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**edit_run_submission**(*payload*, *run_id*)

Edit Run

Edit Run

- Sample Call to edit:

```
curl -X PATCH http://localhost:5000/runs/6/edit
     -H "Authorization: Bearer $TOKEN"
     -F "form-run-name=Updated Curl Run Patch"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 401
{
    "description": "401: Authorization header is expected.",
    "error": 401,
    "message": "Unauthorized",
    "success": false
}
```

app.**run_submission**(*payload*, *run_id*)

Exec Run

Run ML Training based upon run record attributes

- Sample Call to display:

```
curl -X GET http://localhost:5000/train/1
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
HTTP Status Code: 200
<!doctype html>...</html>
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found: The requested URL... try again.",
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

app.**download**(*payload*, *run_id*)

download results file Run ML Training based upon run record attributes

- Sample Call to display:

```
curl -X GET http://localhost:5000/train/1/download
     -H "Authorization: Bearer $TOKEN"
```

- Expected Success Response:

```
HTTP Status Code: 200
File Download. Example:
    PassengerId,Survived
    892,0
    893,0
    894,1
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
 "description": "404 Not Found: The requested URL... try again.",
 "error": 404,
 "message": "Not Found",
 "success": false
}
```

CHAPTER

THREE

ROBOT CLASSIFY'S API MODEL

Introduction

There are three models: - Venue - Artists - Shows

class models.**Project**(***kwargs*)

An ML Project. Projects are the top-orginizing layer for running ML problems

insert()

Inserts a new model into a database. The model must have a unique id and title.

EXAMPLE:

```
p = Project()
p.insert()
```

delete()

deletes a new model into a database the model must exist in the database

EXAMPLE:

```
p = Project()
p.delete()
```

update()

updates a new model into a database the model must exist in the database

EXAMPLE:

```
p = Project.query.filter(p.id == id).one_or_none()
p.name = 'Regression Test'
p.update()
```

class models.**Run**(***kwargs*)

insert()

Inserts a new model into a database. The model must have a unique id and title.

EXAMPLE:

```
p = Project()
p.insert()
```

delete()

deletes a new model into a database the model must exist in the database

EXAMPLE:

```
p = Project()
p.delete()
```

update()
updates a new model into a database the model must exist in the database

EXAMPLE:

```
p = Project.query.filter(p.id == id).one_or_none()
p.name = 'Regression Test'
p.update()
```

CHAPTER

FOUR

ML LIB

4.1 Introduction

The projects module is is a high-level library to process ML jobs at the project level. All interactions can happen with this API

- Manage Projects
- Load Data
- Explore Data
- Auto-execute ML jobs
- Create cleaning rules
- Train the data, finding the best model
- Deploy model, to process ML transactions

mlLib.project.**autoFlaskEvaluateClassifier**(*projectName=None, trainingFile=None, testingFile=None, trainingFileDF=None, testingFileDF=None, targetVariable=None, key=None, predictSetOut=[None], trainingFileOut=None, logFileOut=None, transcriptFile=None, predictFileOut=None, resultsFile=None, modelList=None, confusionMatrixLabels=[], scoring='f1', useProba=False, bottomImportancePrecentToCut=None, setProjectGoals={'f1': (0.9, '>')}, runVerbose=0, recommendOnly=None, basicAutoMethod=None, doExplore=True, doTrain=True, doPredict=True, skewFactor=None, toTerminal=True*)

autoFlaskEvaluateClassifier

Auto-process an ML job for Flask Servers

XXXXX

- Sample Call:

```
Example Call
```

- Expected Success Response:

```
Example Success Response
```

- Expected Fail Response:

```
Example fail response
```

mlLib.project.**autoEvaluateClassifier**(*projectName=None, trainingFile=None, testingFile=None, targetVariable=None, key=None, predictSetOut=[None], trainingFileOut=None, logFileOut=None, transcriptFile=None, predictFileOut=None, resultsFile=None, modelList=None, confusionMatrixLabels=[], scoring='f1', useProba=False, bottomImportancePrecentToCut=None, setProjectGoals={'f1': (0.9, '>')}, runVerbose=1, recommendOnly=None, basicAutoMethod=None, doExplore=True, doTrain=True, doPredict=True, skewFactor=None, toTerminal=True*)

autoEvaluateClassifier

Auto-process an ML job for

XXXXX

- Sample Call:

```
Example Call
```

- Expected Success Response:

```
Example Success Response
```

- Expected Fail Response:

```
Example fail response
```

class mlLib.project.**mlProject**(*name*, *description=None*)

mlProject is the top level object for training and running a ML project. Various object mothods are used to load, review, and train the data, as well as manage running predictions

Example: project = mlProject('Customer Segements', 'clustering model should factor in both aggregate sales patterns and specific items purchased')

setTrainingPreferences(*crossValidationSplits=None*, *parallelJobs=None*, *modelType=None*, *modelList=None*, *testSize=None*, *randomState=None*, *uniqueThreshold=None*, *dropDuplicates=None*, *clusterDimensionThreshold=None*, *varianceThreshold=None*, *kmeansClusters=None*, *useStandardScaler=None*, *fbeta=None*, *runHyperparameters=None*, *runEstimatorHyperparameters=None*, *runMetaClassifier=None*, *runAutoFeaturesMode=None*, *smallSample=None*, *highDimensionality=None*, *gridSearchVerbose=None*, *gridSearchScoring=None*, *featuresToReport=None*, *logTrainingResultsFilename=None*, *useProbaForPredict=None*, *recommendOnly=None*, *basicAutoMethod=None*, *competitionMode=None*, *skewFactor=None*, *bottomImportancePrecentToCut=None*)

setTrainingPreferences

setTrainingPreferences for an ML job

XXXXX

- Sample Call:

```
Example sample call
```

- Expected Success Response:

```
Example Success Response
```

- Expected Fail Response:

```
Example fail response
```

setHyperparametersOverride(*modelName*, *override*, *forBaseEstimator=False*, *forMetaClassifier=False*)

Purpose: Set the hyperparameters to override the defaults for a model

Example:

```
hyperparameters = {
        'lasso__alpha' : [0.001, 0.01, 0.1, 1, 5, 10]
        }
project.setHyperparametersOverride(self, 'lasso', hyperparameters)


hyperparameters = {
    'lasso__alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10]
    }
hyperparameters = {
    'ridge__alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10]
}
hyperparameters = {
    'elasticnet__alpha':
        [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10],
    'elasticnet__l1_ratio' : [0.1, 0.3, 0.5, 0.7, 0.9]
}
hyperparameters = {
    'randomforestregressor__n_estimators': [50, 100, 200, 500],
    'randomforestregressor__max_features': ['auto', 'sqrt', 0.33]}
hyperparameters = {
    'gradientboostingregressor__n_estimators': [50, 100, 200, 500],
    'gradientboostingregressor__learning_rate':
            [0.001, 0.05, 0.1, 0.5],
    'gradientboostingregressor__max_depth': [1, 5, 10, 50]}
hyperparameters = {'decisiontreeregressor__max_depth':
                    [1, 8, 16, 32, 64, 200]}
hyperparameters = {
        'logisticregression__C' : np.linspace(1e-4, 1e3, num=50),
        'logisticregression__max_iter': [25, 50, 100, 300, 500]
        }
hyperparameters = {
        'logisticregression__C' : np.linspace(1e-4, 1e3, num=50),
        'logisticregression__max_iter': [25, 100, 300, 500]
        }
hyperparameters = {'randomforestclassifier__n_estimators':
                    [100, 200],
                    'randomforestclassifier__max_features':
                    ['auto', 'sqrt', 0.33]}
hyperparameters = {'gradientboostingclassifier__n_estimators':
```

(continues on next page)

(continued from previous page)

```
                    [50, 100, 200, 500],
    'gradientboostingclassifier__max_depth': [1, 10, 50, 100],
    'gradientboostingclassifier__learning_rate':
    [.1, .01, .001,.0001]}
```

setConfusionMatrixLabels(*list*)

Example: project.setConfusionMatrixLabels([(0,'Paid'), (1, 'Default')])

setTarget(*value*, *boolean=False*, *trueValue=None*, *convertTable=None*, *tableName=None*)
Purpose: Set the target variable for supervised learning.

Call: setTarget(self, value, boolean=False, trueValue=None, convertTable=None, tableName=None):

Example:

project.setTarget('loan_status')

trueValue = what is the true values boolean = is this a boolean value convertTable = a table of how to convert values

importFile(*name*, *type=None*, *description=None*, *location=None*, *fileName=None*, *sheetName=None*, *df=None*, *hasHeaders=False*, *range=None*, *isDefault=False*)

def importFile(self, name, type=None, description=None, location=None, fileName=None, sheetName=None, hasHeaders = False, range=None, isDefault=False):

project.importFile('Loan Data', type='csv', description='Lending Club Data from 2017-2018', fileName='LendingClub2017_2018ready.csv', hasHeaders = True, isDefault=True)

exportFile(*name*, *filename*)
Purpose: Export the named file. (Projects can have multiuple files associated with them)

Call: def exportFile(self, name, filename):

Example: project.exportFile('Loan Data', 'fileout.csv'):

getColumn(*name*, *columnName*)
Purpose: Get a columns from the data file

Call: def getColumn(self, name, column):

Example: project.getColumn('Loan Data','Name')

exploreData(*fileName=None*)

Purpose: Run the explore data function. This will review the data and make recommendations

Call: exploreData(self):

Example: project.exploreData()

initCleaningRules(*fileName=None*)
Before adding any cleaning rules you must init

project.initCleaningRules()

project.addManualRuleForDefault(ed.CLEANDATA_REBUCKET_TO_BINARY, 'term', [['36 months', ' 36 months'], '36'])

project.addManualRuleForDefault('ed.CLEANDATA_REBUCKET_TO_BINARY, 'term', [['60 months', ' 60 months'], '60'])

cleanProject (*fileName=None*)
Purpose: Run the cleaning rules established for a project.

Call: cleanProject(self)

Example: project.cleanProject()

cleanAndExploreProject (*fileName=None*)
Purpose: Run clean and explore together

Call: def cleanAndExploreProject(self)

Example: project.cleanAndExploreProject()

prepProjectByName (*tableName=None*, *outFile=None*)

Purpose: Prepare the 'table' for training. This will one-hot encode, for example.

Call: prepProjectByName(self, tableName=None)

Example: project.prepProjectByName('Loan Data')

writePreppedFileByName (*filename*, *tableName=None*)
Purpose: Once a file has been cleaned and explorred

Call:

Example:

writeTrainingSetFileByName (*filename*, *tableName=None*)
Purpose:

Call:

Example:

trainProjectByName (*tableName=None*)
Purpose:

Call:

Example:

prepProjectByBatch ()
Purpose:

Call:

Example:

trainProjectByBatch ()
Purpose:

Call:

Example:

exportBestModel (*filename*, *tableName=None*)
Purpose:

Call:

Example:

createPredictFromBestModel (*tableName=None*)
Purpose:

Call:

Example:

createPredictFromNamedModel (*namedModel*, *tableName=None*)
Purpose:

Call:

Example:

exportNamedModel (*namedModel*, *filename*, *tableName=None*)
Purpose:

Call:

Example:

addManualRuleForTableName (*tableName*, *functionName*, *columnName*, *value*, *forPredict=True*)
Purpose:

Call:

Example:

addManualRuleForDefault (*functionName*, *columnName=None*, *value=None*, *forPredict=True*)
Purpose:

Call:

Example:

setGoals (*goals*)
Purpose:

Call:

Example:

project.setGoals({'AUROC':(0.70,'>'),'Precision':(0.386,'>'), 'fbeta':(0.44,'>')})

setOngoingReporting (*True*, *'Loan Data'*)

displayAllScores (*'Loan Data'*)
def displayAllScores(self, fileName):

reportResultsOnTrainedModel (*fileName*, *modelName*)
Purpose:

Call:

Example:

showFeatureImportances (*fileName*, *modelName*)
Purpose:

Call:

Example:

logTrainingResultsRunDescription (*description='None'*)
Purpose:

Call:

Example:

logTrainingResults (*fileName*, *outputFileName*, *inputModelName=None*)
Purpose:

Call:

Example:

class mlLib.project.**predictProject**(*project*, *tableName=None*, *namedModel=None*)
Purpose: predictProject

Call:

Example:

importPredictFile(*name*, *type=None*, *description=None*, *location=None*, *fileName=None*, *sheetName=None*, *hasHeaders=False*, *range=None*)
Purpose:

Call:

Example:

importPredictFileFromProject(*project*, *tableName*)
Purpose:

Call:

Example:

importPredictFromDF(*df*, *readyForPredict=False*)
Purpose:

Call:

Example:

prepPredict()
Purpose:

Call:

Example:

exportPreppedFile(*filename*, *columnName=None*, *columnData=None*, *columnName2=None*, *columnData2=None*)
Purpose:

Call:

Example:

getColumn(*columnName*)
Purpose: Get a columns from the data file

Call: def getColumn(self, column):

Example: prdict.getColumn('Name')

exportPredictClass(*filename*)
Purpose:

Call:

Example:

addToPredictFile(*columnName*, *columnData*)
Purpose:

Call:

Example:

removeFromPredictFile(*columns*)
Purpose:

Call:

Example:

keepFromPredictFile(*columns*)
Purpose:

Call:

Example:

exportPredictFile(*filename*)
Purpose:

Call:

Example:

getPredictFileDF()
Purpose: Return a datraframe of the predict file.

Call:

Example:

runPredict()
Purpose:

Call:

Example:

mlLib.project.**loadPredictProject**(*filename*)
Purpose:

Call:

Example:

mlLib.project.**plot_confusion_matrix**(*cm*, *classes*, *normalize=False*, *title='Confusion matrix'*, *cmap=<matplotlib.colors.LinearSegmentedColormap object>*)
Purpose:

Call:

Example:

This function prints and plots the confusion matrix. Normalization can be applied by setting *normalize=True*.

mlLib.project.**makeStack**(*classifier*, *list*, *alias=None*)
Purpose:

Call:

Example:

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

m

INDEX