## ADTA5340 Final Project

## PART I: A Strategy to Employ Machine Learning in a Firm

### Define data to collect

First thing we need to determine what data we would collect. We need to take samples of datasets and estimate their current volume and growth. We need to check if data requires additional preprocessing. We need to plan for enough storage and processing power and we should be able to manage capacity at real-time without affecting processing capabilities.

The system should not be used as data store for any other business critical application, rather it should be able to collect / consume data from those systems.

### High-level architecture

The system would have several independent layers of processing. Independent levels would ensure resiliency and provide ability to scale each layer independently.

#### Data collection layer.

This layer would be used to collect and store initial data from other systems. This layer should have enough storage capacity to collect data for a period of time before it would be moved to other layers and processed there. Capacity of this system should be kept at maximum of 50%. Depends on how much data coming each second, we would need to create a storage with fast write.

#### Data preprocessing layer.

The purpose of this layer is to verify, clean, and get data ready for further processing. Also, this layer would be used as a quality assurance system. If there would be large amount of preprocessing required for some dataset, we would be able to alert data quality issues to respective systems’ owners.

#### Data storage layer.

This is long-term storage. It would require high capacity and fast read time. We should be able to increase capacity fast without affecting currently stored data.

#### AI/ML processing layer.

This layer requires highest number of CPU and memory to constantly build and run models. Storage requirements are not high, it should have some storage to keep some temporary files, but most of the time it should be able to keep everything in memory.

#### Reporting layer.

This is separate reporting layer. We need it to generate and store reports. In terms of CPU, memory and storage it would have mediocre requirements.

#### Web Access layer.

We need this layer to perform following functions:

* Overall system management, including access management;
* User access to run models and get results;
* Provide access to stored report and to report management functions;

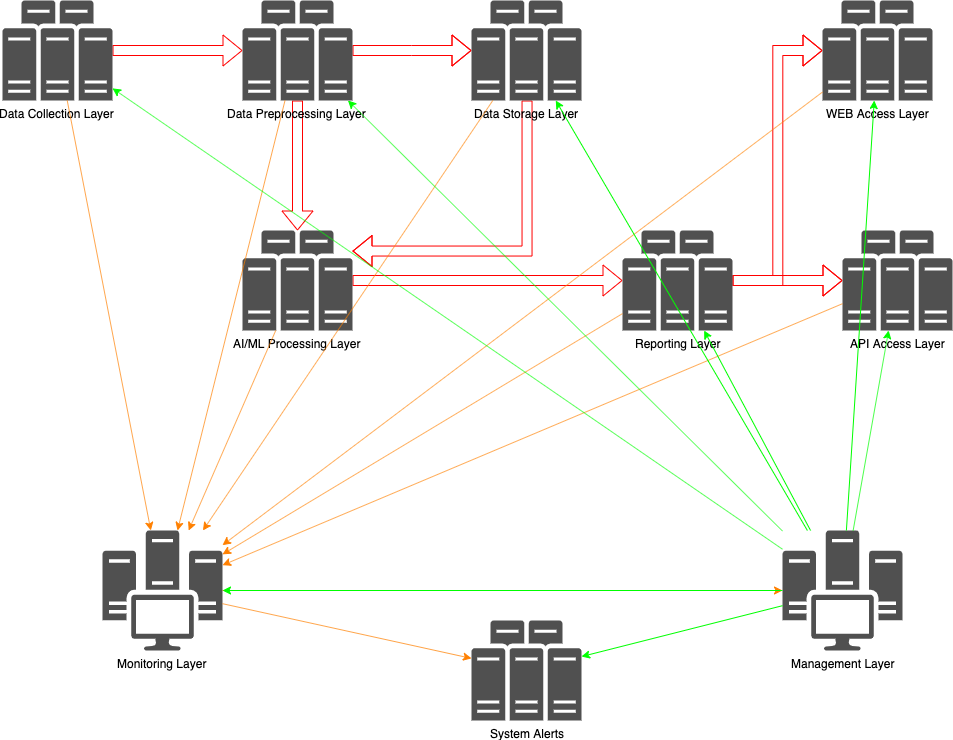
#### API Access layer.

Modern organizations are data driven. Lot of systems produce, process, and store different data, that consumes by other systems in automated manner. We need a layer that could provide data for other machines in a machine-readable way.

#### System Alerts layer.

This layer is required to send alerts related to the system itself (health, reports readiness, security events, etc.), as well as alerts related to various external systems (i.e. data quality)

### System diagram



### Technologies to be used.

#### Hardware

Hardware should be based on commodity systems of x86/x64 architecture. This would provide ability to replace systems fast with low cost. Different components should be packed into standard racks with ability to add new racks into the system.

Also, we should replace components by racks. When certain percent of systems within a rack have issues would should replace the entire rack. System should be able to rebalance itself automatically.

#### Software

Software should be mostly based on open-source software. It would provide ability to both: change code of applications or develop new code, and at the same time use achievements from other software developers.

## PART II: Big Data, Artificial Intelligence, and Machine Learning

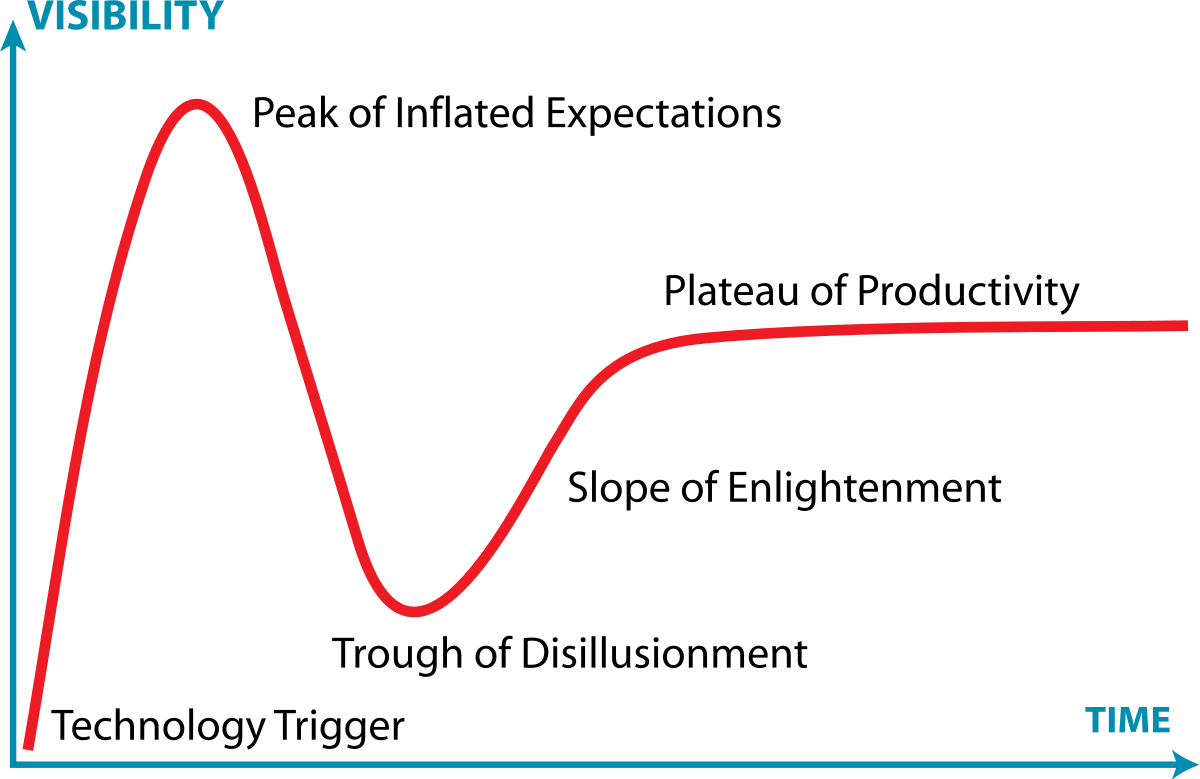
### The history of artificial intelligence until now

The term Artificial Intelligence was created by John McCarty, Marvin Minsky, Allen Newell and Herbert Simon in 1956 during a workshop at Dartmouth College. As with any new technology, AI went through different phases. When initial idea appears few early adopters or evangelists promoted it with promises like it would be a “silver bullet” for world problems. Some governments (USA and UK) and private companies started to invest money into it.

Lot of theories and related programs were developed during this phase:

* Reasoning as Search when search algorithm goes step by step through possible solutions and combinations and, when hitting a dead-end (no solution found), returns back (or backtracking) and tries another path. Researches tried to find a way for an algorithm to detect paths that would not have a solution, otherwise it need to go over huge number of combinations before getting an adequate solution.
* Natural Language tried to teach computer to speak with each other and understand each other using human language (i.e. English)
* Micro-Worlds idea was proposed by Marvin Minsky. He proposed that AI should focus on small objects like physics focus on basic forms and use these small objects to build larger constructions.

Unfortunately, due to lack of results (because there was no enough computing power, theories just started developing) investments stopped soon, and AI was put aside. But it was not forgotten. Universities continued researches in this area, new theories appeared and were tested in academic environments.



Picture 1. Hype cycle (author: Jeremykemp at English Wikipedia)

The second appearance of AI started in late 1970-s, beginning of 1980-s when Japan government first and then US and UK governments and some corporations started invest money into AI again. It was time for expert systems and theories around knowledge. The general idea was to put expert knowledge into a computer system, and let people ask this system questions. The limitation was that such systems were limited to only specific, small areas of knowledge. Also, these systems used very expensive hardware. In late 80-s and early 90-s personal computers from Apple, IBM, and other took over the computer market. Special expert systems were too costly to maintain and support, they have issues with data quality, and were able to support just very narrow fields of knowledge, so money flows stopped again.

Only in the beginning of 2000-s private companies got enough computing resources, and AI started to show its abilities in different areas. In 1997 the Deep Blue computer by IBM won a chess match against Garry Kasparov, world chess champion at the time. Different DARPA challenges in 2000-s showed that computers could navigate in a desert or in a city.

The most recent example is a computer that won a GO game against a world champion. The most interesting fact is that during learning phase this computer played against its twin. Computers were able to develop tactics that humans never thought about.

Lot of companies invest into AI/ML technologies now. IT giants, financial conglomerates, governments, everybody expect AI/ML would benefit them. Some companies embed ML tools into their products to help aging population to live longer and fuller lifes.

Modern AI is continue developing. Deep Machine Learning, Big Data, and General Artificial Intelligence are the main focuses of research now.

### Select three different sectors of the U.S. economy, do research, and discuss the impacts of **big data** and **machine learning** on **each** of them

Machine Learning (ML) affects all sectors of the US economy: from financial institutions to video streaming services, from Internet Commerce to non-profit organizations. All of them benefit from artificial intelligence (AI) and ML. Some of them has more benefits right now, but situation will be different very soon, when technology would be adopted to all companies, of any size and any industry.

Internet giants such as Google, Amazon, Facebook and Microsoft are obvious leaders and beneficiaries of AI/ML technologies. They use these technologies to increase their main revenue stream which is advertisement. In early days on the Internet, everybody hated ad banners. Ads were not relevant to a person who browsed a site.

Google was the first to understand the potential of precise profiling of an Internet user. They started to collect a lot of data through their analytics service. Initially results were not good. Google had a lot of collected data, but banners still were not relevant to search results or visited sites, so they started to develop technologies to process them. They know that they need to use statistical science to get the results they need, but with amount of data they collected they needed new ways to automate processing of this data. And they started to develop tools for that.

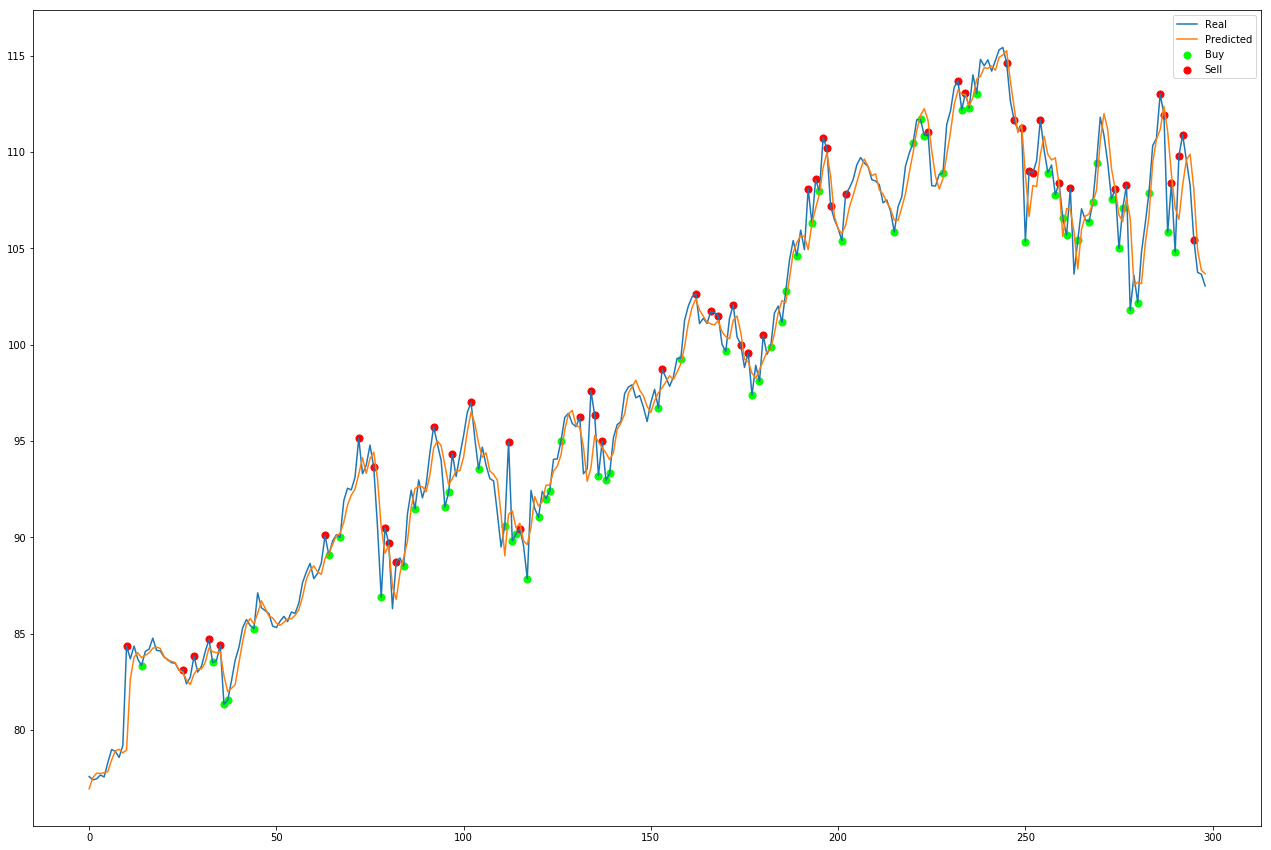
At the same time Google’s team understood that correct answer to a search request is no less important that proper ad in proper time. If a user would have ability to ask something in natural language and would provide adequate results, that would be a benefit for a search engine. The better search results, the more users use it, the more training data search engine has, the more profits from advertisement.

But not only Google started to benefit from AI/ML, other IT companies discovered the power of AI/ML too. Microsoft, Amazon, Facebook, Yandex, and other uses AI/ML to provide value to their users and at the same time guide money into these companies.

Google started to use AI/ML to process health data. IBM’s Watson provides answers for growing number of areas where sometimes it could beat experts in speed and quality of answers. Yandex uses AI/ML to teach a car to drive on busy streets of Russian capital.

But not only IT companies are big players within AI/ML. Financial industry uses AI/ML to predict futures’ prices, or protect their customers from card fraud. Did you have a call from a bank’s call-center to notify you that your card was used in suspicious transaction(s)? I did.

Banks have so much information about their customers, so they could predict customers spending patterns or detect activity that does not correspond to their customer’s behavior. Banks could offer you extension of a credit line when you need it, or give you an offer that would be very hard to decline.

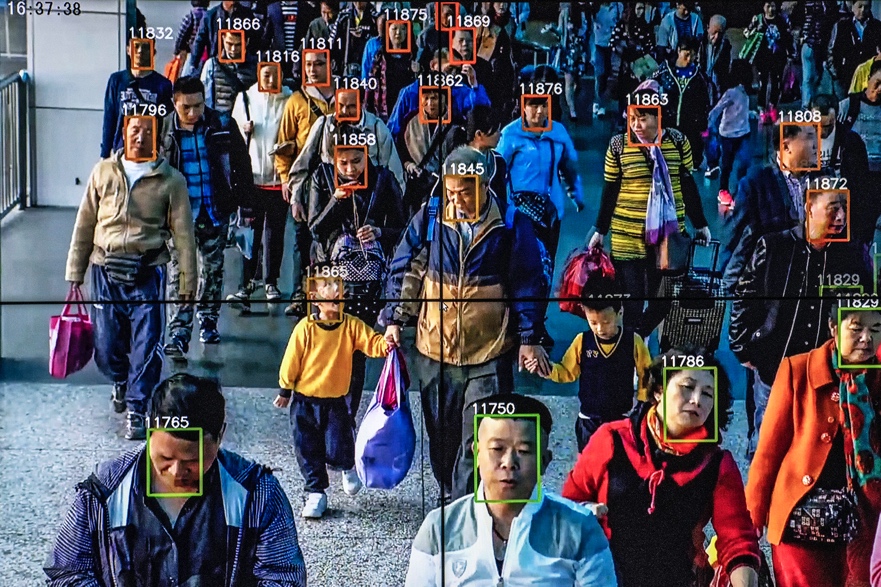


<https://towardsdatascience.com/getting-rich-quick-with-machine-learning-and-stock-market-predictions-696802da94fe>

High frequency trading is another area where banks use AI/ML. An algorithm sells or buys something with such a speed that no human trader could do. But this algorithm also needs news from around the world to make decisions. An there we would have another algorithm that collects news and predicts how stocks, bonds, or currencies would react. Automated “trader” uses these predictions to do such trades.

Another example is collection of information about companies that would go to IPO or would like to merge with another company, etc. Earlier people would collect and combine these sources of information together, now algorithms do it. Algorithms could predict that a company wants to do an IPO even before official news break silence.

Governments also uses AI/ML for their profit. Face recognition systems at some airports reduce load for TSA officers. Video surveillance systems could detect suspects on a street in a crowd. Sometimes these systems could be easily fooled, but they are still in developing phase. Pretty soon they would be able to recognize a person with almost 100% probability. And while it seems the world moves towards a “Big Brother” future, I hope such systems would not be abused to control regular people.



Where’s Waldo? Monitors at the Beijing offices of A.I.-software startup Megvii play a video showing how its facial recognition software works.

Gilles Sabrie—The New York Times/Redux

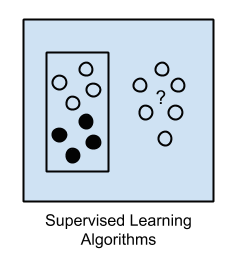
<https://fortune.com/2018/10/28/in-china-facial-recognition-tech-is-watching-you/>

Militaries around the world also see opportunities in ML. From detecting enemy movements to removing your own soldiers from a battle field to safe place thousand miles away from it, these technologies will take their place. Just hope humanity would not be killed by some AI that decided the Earth is better without humans.

### Discuss **in detail** the three major styles of learning in machine learning: (1) Supervised Learning, (2) Unsupervised Learning, and (3) Semi-Supervised Learning

There are three primary learning styles for AI/ML: supervised, unsupervised, and semi-supervised learning.

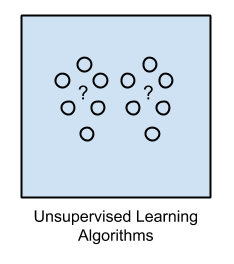
For supervised learning training dataset has marked by an operator (a person who is going to train a model), the operator knows meanings of all variables and understands dependent and independent variables.



Supervised learning works with relatively small datasets, where independent and dependent variables could be clearly identified, data in a dataset could be effectively verified and cleaned.

Examples for supervised learning algorithms are linear regressions, decision trees (regression and classification).

For unsupervised learning, dataset has no labels or markers for output data, algorithm learns itself. It learns to find common dependencies or tries to understand general rules on how to separate the data into segments.

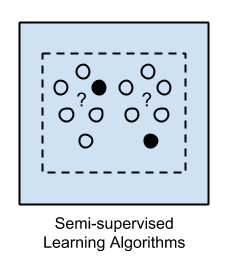


Usually datasets for unsupervised learning are very large and hard to clear. Imagine a collection of videos on the Youtube or digital images’ libraries. If you do not have enough description in metadata it would be hard to find something with a search request like: “show me a list of all images with cats”. The only way is to train ML model to detect a cat on an image, them use this model to find cats on images.



Another example for unsupervised learning model would be detection of obstacles on a road. We could train algorithm to find/detect an obstacle on a video stream, an operator would be unable to tell in advance if there is an obstacle, algorithm should learn itself to detect it.

Semi-supervised learning is combination of the two techniques above. We could label some data (inputs and outputs) in the dataset but not for the whole dataset. ML algorithm should learn itself what is it and how to mark it. But with even small amount of labeled data algorithm usually provides more accurate results than completely unsupervised learning.



An example for semi-supervised learning could be detection of traffic lights. An operator would mark some outputs for a ML algorithm like shape of traffic lights, how they usually mounted, what different colors mean, etc. But the operator could not mark all possible variants of traffic lights setups, algorithm would use information it has to detect and properly interpret information from images or video streams.

## PART III: Data Preprocessing

For data preprocessing task I found a dataset “Aviation Data and Documentation from the NTSB Accident Database System”.

URL for the dataset: <https://catalog.data.gov/dataset/aviation-data-and-documentation-from-the-ntsb-accident-database-system>

Public: This dataset is intended for public access and use.

License: No license information was provided. If this work was prepared by an officer or employee of the United States government as part of that person's official duties it is considered a U.S. Government Work.

This dataset was initially in XML format, I had to export it into CSV to work with it. The dataset has 84088 records and 31 variables:

\_EventId - 0 - Event identifier, each event should have an id. Field is not useful for further analysis as it is unique.

\_InvestigationType - 4 - Type of investigation (Accident, Incident). Some events have no type. NULL value could be replaced with Unknown.

\_AccidentNumber - 0 - Number (identification) of an event in some recording systems. Field is not useful for further analisys as it is unique.

\_EventDate - 0 - Date of an event. Cannot be NULL.

\_Location - 76 - Location of an event (city, state in case of the US or other identifiable location if outside of the US). Some locations are NULL. There are other variables for locations data.

\_Country - 507 - some locations do not have county filled. It could have NULL values when an event happened outside of any country (i.e. under an ocean or a sea).

*\_*Latitude – 54039

\_Longitude – 54048

\_Latitude and \_Longitude are geographical coordinates of an event. It should not be empty, but there are other location variables that could substitute them. If we really need them we need to work with other variables to get exact coordinates of an event.

\_AirportCode - 36439 - each airport has a code (3 or 4 letters). An event could happen outside of airport, so it could have NULL values. This variable could be used to define geographic location of an event.

\_AirportName - 33735 - Each airport has a human readable name. In case an event of outside of any airport this field would have NULL value. I would ignore Location variables. An event could happen at any location. But for general statistics these values could be useful to find which locations have more incidents.

\_InjurySeverity - 0 - Each event has assigned severity. Cannot be NULL.

\_AircraftDamage - 2676 - Describes damage to an aircraft. NULL value indicates no damage, NULL should be replaced with "NODAMAGE".

\_AircraftCategory - 56751 - Describes category of an aircraft like airplane, balloon, etc. NULL value indicates that category was not defined or not recorded. Missing value could be determined by Make and \_Model variables.

\_RegistrationNumber - 3778 - Categoriacal value (non-numeric). NULL value indicates that aircraft was not registered. For analysis I would remove this column. Replacing NULL with something like "NOT REGISTERED" would affect results of analysis: lot of events would be falsely attributed to it.

\_Make - 70 - Each aircraft should have a producer like Boeing, Airbus, Cessna, etc. But some are made by people and do not have registered Maker.

\_Model - 99 - Some aircrafts do not have models because they were built by private persons.

\_AmateurBuilt - 592 - indicates if aircraft is built by amateur builder. NULL value indicates that data was not recorded for this variable, NULL could be replaced with "NO"

\_NumberOfEngines - 4970 - Some aircrafts has no engines (like baloon or glider). But sometimes aircraft definetely have 1 or more engines but information was not collected.

\_EngineType - 4280 - Describes engine types. NULL indicates either there is no an engine, or data is missing.

\_FARDescription - 57056 - Describes if an aircraft performed a specific function (i.e. Armed Forces or General Aviation, etc.). NULL indicates no data is recoded, it should be changed to Unknown.

\_Schedule - 72269 - Indicates if a flight was scheduled (like AA20 from DFW to London Heathrow), unscheduled (i.e. for Armed Forces interception) or unknown. NULL values should be replaced with "NO".

\_PurposeOfFlight - 4791 - Describes the purpose of flight. NULL values should be replaced by "Unknown".

\_AirCarrier - 79927 - Names air carrier. Most of events have no air carrier, because event happened with personal aircrafts that do not perform regular passengers’ transfers. I would ignore this column.

\_TotalFatalInjuries - 27068 - Describes number of total fatal injuries. NULL values should be replaced with "0".

\_TotalSeriousInjuries - 29726 - Describes number of total serious injuries. NULL values should be replaced with "0"

\_TotalMinorInjuries - 28523 - Describes number of total minor injuries. NULL values should be replaced with "0"

\_TotalUninjured - 14507 - Describes number of total uninjured. NULL values show that there are no data. Normally it should show total people on board minus all injures combined. But there is no variable for total passengers, so we cannot calculate this value. I would ignore it.

\_WeatherCondition - 3060 - This variable describes weather conditions. NULL values indicate that there is no data on file, or weather had nothing to do with an event.  
BroadPhaseOfFlight - 6691 - Describes phase of flight when an event happened. NULL values should have "UNKNOWN"  
ReportStatus - 0 - Describes status of a report. Cannot have NULL values.  
\_PublicationDate - 14236 - Date of a report publication. NULL value shows that report is not published (could be "Secret" or not ready for publication)

Step 1. I reduced number of variables for this dataset:

df2 = df[[**"\_InvestigationType"**, **"\_EventDate"**, **"\_InjurySeverity"**, **"\_AircraftDamage"**, **"\_Make"**, **"\_Model"**, **"\_AmateurBuilt"**, **"\_FARDescription"**, **"\_Schedule"**,**"\_PurposeOfFlight"**, **"\_TotalFatalInjuries"**, **"\_TotalSeriousInjuries"**, **"\_TotalMinorInjuries"**, **"\_BroadPhaseOfFlight"**]]

Step 2. I replaced NaNs with proper values:

df2[[**'\_TotalSeriousInjuries'**, **'\_TotalMinorInjuries'**, **\_TotalFatalInjuries'**]] = df2[[**'\_TotalSeriousInjuries'**, **'\_TotalMinorInjuries'**, **'\_TotalFatalInjuries'**]].replace(np.NaN,0)  
df2[[**'\_InvestigationType'**]]=df2[[**'\_InvestigationType'**]].replace(np.NaN,**"Unknown"**)  
df2[[**'\_AircraftDamage'**]]=df2[[**'\_AircraftDamage'**]].replace(np.NaN,**"No damage"**)  
df2[[**'\_Make'**]]=df2[[**'\_Make'**]].replace(np.NaN,**"No make"**)  
df2[[**'\_Model'**]]=df2[[**'\_Model'**]].replace(np.NaN,**"No model"**)  
df2[[**'\_AmateurBuilt'**]]=df2[[**'\_AmateurBuilt'**]].replace(np.NaN,**"No"**)  
df2[[**'\_FARDescription'**]]=df2[[**'\_FARDescription'**]].replace(np.NaN,**"No"**)  
df2[[**'\_Schedule'**]]=df2[[**'\_Schedule'**]].replace(np.NaN,**"NO"**)  
df2[[**'\_PurposeOfFlight'**]]=df2[[**'\_PurposeOfFlight'**]].replace(np.NaN,**"Unknown"**)  
df2[[**'\_BroadPhaseOfFlight'**]]=df2[[**'\_BroadPhaseOfFlight'**]].replace(np.NaN,**"UNKNOWN"**)

Step3. Checked that there are no more NaNs.

print(df2.isnull().sum())

## PART IV: Machine Learning: Supervised

The dataset abalone.csv has 4177 records for 9 variables. These variables are:

1. Sex

2. Length: mm: Longest shell measurement

3. Diameter: mm : perpendicular to the length

4. Height : mm : with meat in the shell

5. Whole weight : grams : whole abalone

6. Shucked weight : grams : weight of meat

7. Viscera weight : grams : gut weight (after bleeding)

8. Shell weight : grams : after being dried

9. Rings : integer : +1.5 gives the age in years

After loading the dataset into Jupyter notebook, data quality was checked to ensure there are no NaN values, or values that need to be encoded from Categorical into Numeric type.

print(df.head(5))

Sex Length Diameter Height Whole Weight Shucked Weight Viscera Weight \

0 M 0.455 0.365 0.095 0.5140 0.2245 0.1010

1 M 0.350 0.265 0.090 0.2255 0.0995 0.0485

2 F 0.530 0.420 0.135 0.6770 0.2565 0.1415

3 M 0.440 0.365 0.125 0.5160 0.2155 0.1140

4 I 0.330 0.255 0.080 0.2050 0.0895 0.0395

Shell\_Weight Rings

0 0.150 15

1 0.070 7

2 0.210 9

3 0.155 10

4 0.055 7

print(df.isnull().sum())

Sex 0

Length 0

Diameter 0

Height 0

Whole Weight 0

Shucked Weight 0

Viscera Weight 0

Shell\_Weight 0

Rings 0

dtype: int64

print(df.groupby('Sex').size())

Sex

F 1307

I 1342

M 1528

dtype: int64

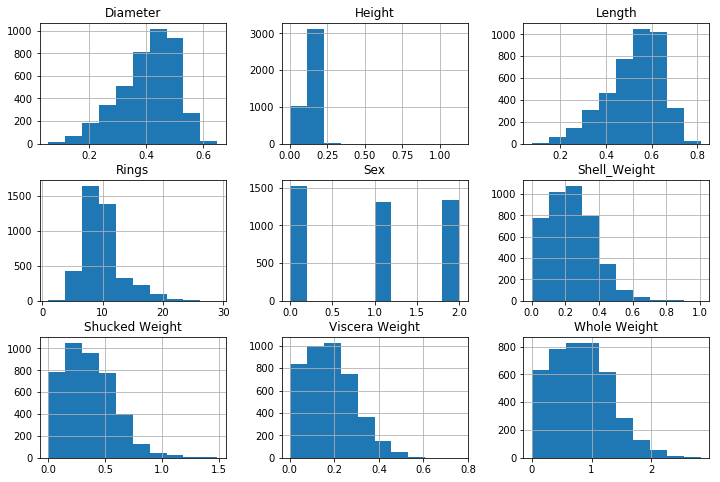
Variable Sex needed to be encoded into Numeric form:

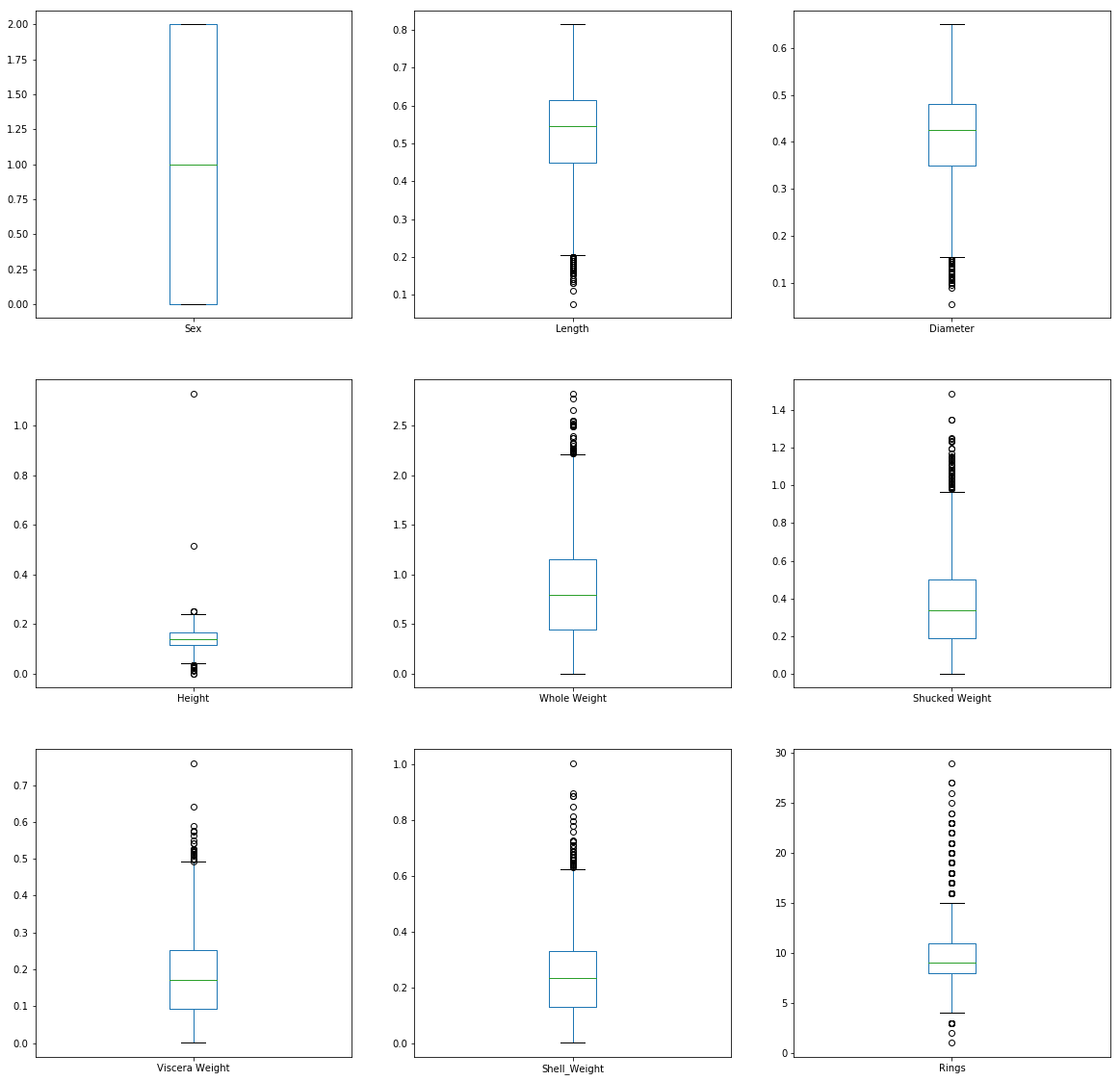
df[['Sex']]=df[['Sex']].replace("M",0)

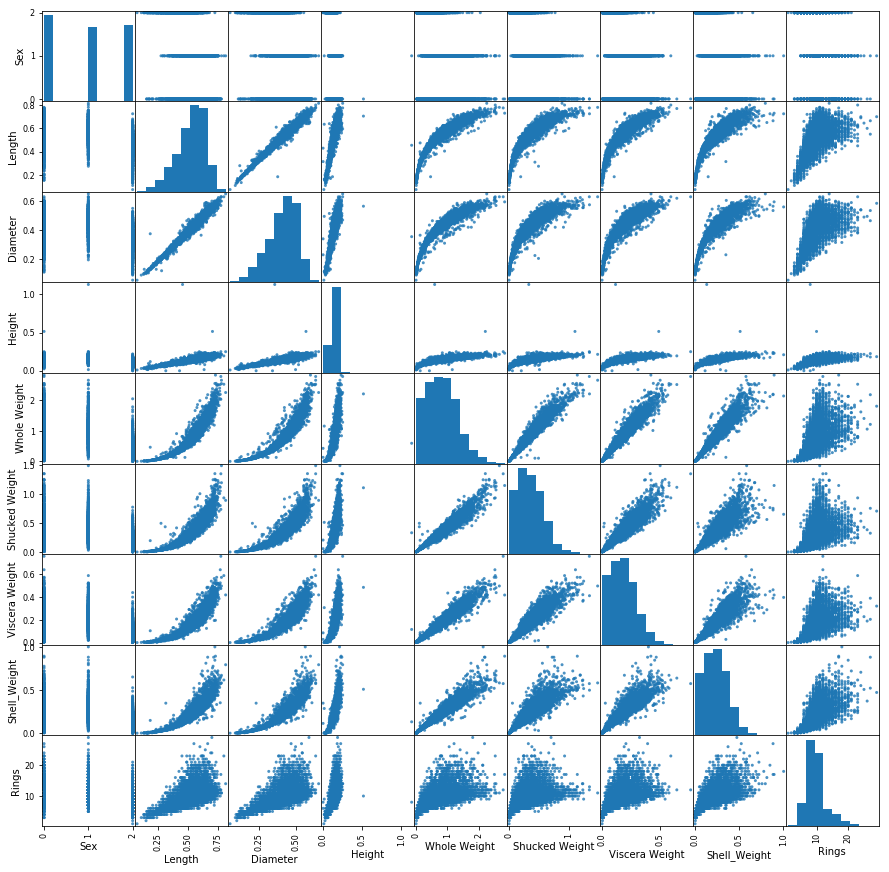
df[['Sex']]=df[['Sex']].replace("F",1)

df[['Sex']]=df[['Sex']].replace("I",2)

After encoding of data EDA was performed: few different kinds of graphs were created to explore data: Histogram, S-Boxes, Scatter plots.







Histograms for Diameter and Length are skewed left but after reaching the top, values rapidly drop. Weight variables histograms are skewed right. Rings histogram shows that numbers of data points rapidly grow to a range 8-10, but then they drop (more skewed right) to range 20+.

S-Boxes for Length and Diameter show lot of outliers below min values, other variables show high number of outliers above max values.

Scatter plots shows that Females live slightly longer than Males, and I-sex have slightly shorter life.

For ML algorithm I chose Linear Regression model: we have continuous numeric variables, except for the Sex variable which is Categoric. Also, if we look at scatter plots, we could see that Linear Regression model fits very well.

Initial dataset was separated into train and test sets, then I used train set to train the model, and test set to control results. After training I got a R-Squared error 0.533089828830458.

For prediction I created two new values:

print("Age in years:",model.predict([[0, 0.245, 0.300, 0.100, 0.4578, 0.3333, 0.500, 0.1223]])**+**1.5)

Age in years: [2.52765714]

print("Age in years:",model.predict([[1, 0.442, 0.100, 0.400, 0.6778, 0.2333, 0.670, 0.3456]])**+**1.5)

Age in years: [6.57962604]

As model predicts number of rings for a shell we have to add 1.5 to the results to get Age in Years.

K-Fold evaluation used to evaluate the model gave following results as NME: -5.217558286600992.

## PART V: Machine Learning: Supervised

Dataset adult\_salary.csv has 48842 records and 15 variables. Most of string values has a space symbol at the beginning. With Excel I run a replacement function to remove all extra space symbols. Then I checked if there were any NaN values. There were none.

The dataset has a variable Fnlwgt that was added as a part of data collection and represent no value to the analysis. I removed it with:

df2 = df[[**"Age"**,**"Emp\_type"**,**"Education"**,**"Education\_num"**,**"Marital"**,**"Occupation"**,**"Relationship"**,  
 **"Race"**,**"Sex"**,**"Capital\_gain"**,**"Capital\_loss"**,**"weekly\_hours"**,**"Country"**,**"Income"**]]

Then for Classification algorithms I had to replace all string values with integers.

df2[[**'Income'**]]=df2[[**'Income'**]].replace(**"<=50K."**,0)  
df2[[**'Income'**]]=df2[[**'Income'**]].replace(**">50K."**,1)  
df2[[**'Income'**]]=df2[[**'Income'**]].replace(**"<=50K"**,0)  
df2[[**'Income'**]]=df2[[**'Income'**]].replace(**">50K"**,1)  
Sex = Series([0,1],index=[**'Male'**,**'Female'**])   
df2[**'Sex'**]=df2.Sex.map(Sex)  
df2.Emp\_type = pd.Categorical(df2.Emp\_type)  
df2[**'Emp\_type'**] = df2.Emp\_type.cat.codes  
df2.Education = pd.Categorical(df2.Education)  
df2[**'Education'**] = df2.Education.cat.codes  
df2.Marital = pd.Categorical(df2.Marital)  
df2[**'Marital'**] = df2.Marital.cat.codes  
df2.Occupation = pd.Categorical(df2.Occupation)  
df2[**'Occupation'**] = df2.Occupation.cat.codes  
df2.Relationship = pd.Categorical(df2.Relationship)  
df2[**'Relationship'**] = df2.Relationship.cat.codes  
df2.Race = pd.Categorical(df2.Race)  
df2[**'Race'**] = df2.Race.cat.codes  
df2.Country = pd.Categorical(df2.Country)  
df2[**'Country'**] = df2.Country.cat.codes

After these modifications I got a dataset with only numerical values:

print(df2.dtypes)  
Age int64

Emp\_type int8

Education int8

Education\_num int64

Marital int8

Occupation int8

Relationship int8

Race int8

Sex int64

Capital\_gain int64

Capital\_loss int64

weekly\_hours int64

Country int8

Income int64

dtype: object

EDA Analysis showed that Categorical values gave big standard deviation errors with exception of Age and Weekly hours variables. Only these two are continuous numeric variables in the dataset.

Age Emp\_type Education Education\_num Marital \

count 48842.000000 48842.000000 48842.000000 48842.000000 48842.000000

mean 38.643585 3.870439 10.288420 10.078089 2.618750

std 13.710510 1.464234 3.874492 2.570973 1.507703

min 17.000000 0.000000 0.000000 1.000000 0.000000

25% 28.000000 4.000000 9.000000 9.000000 2.000000

50% 37.000000 4.000000 11.000000 10.000000 2.000000

75% 48.000000 4.000000 12.000000 12.000000 4.000000

max 90.000000 8.000000 15.000000 16.000000 6.000000

Occupation Relationship Race Sex Capital\_gain \

count 48842.000000 48842.000000 48842.000000 48842.000000 48842.000000

mean 6.577700 1.443287 3.668052 0.331518 1079.067626

std 4.230509 1.602151 0.845986 0.470764 7452.019058

min 0.000000 0.000000 0.000000 0.000000 0.000000

25% 3.000000 0.000000 4.000000 0.000000 0.000000

50% 7.000000 1.000000 4.000000 0.000000 0.000000

75% 10.000000 3.000000 4.000000 1.000000 0.000000

max 14.000000 5.000000 4.000000 1.000000 99999.000000

Capital\_loss weekly\_hours Country Income

count 48842.000000 48842.000000 48842.000000 48842.000000

mean 87.502314 40.422382 36.749355 0.239282

std 403.004552 12.391444 7.775343 0.426649

min 0.000000 1.000000 0.000000 0.000000

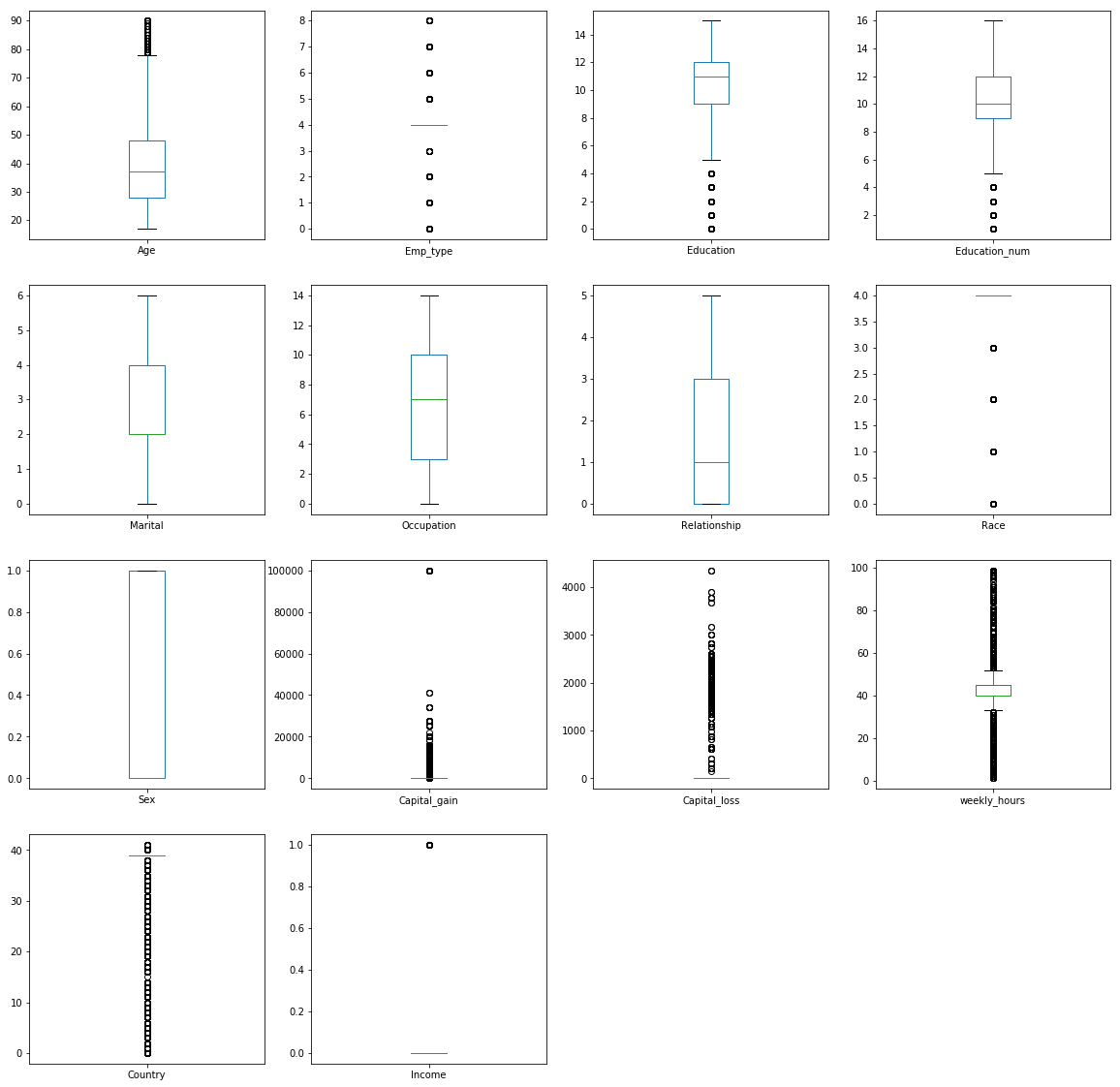
25% 0.000000 40.000000 39.000000 0.000000

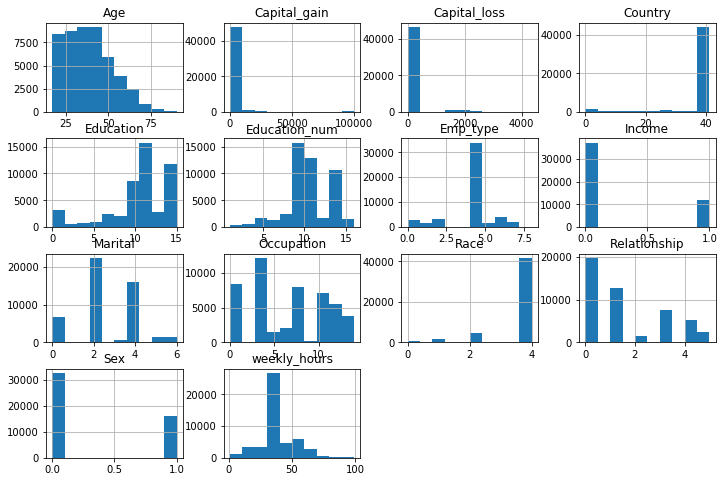
50% 0.000000 40.000000 39.000000 0.000000

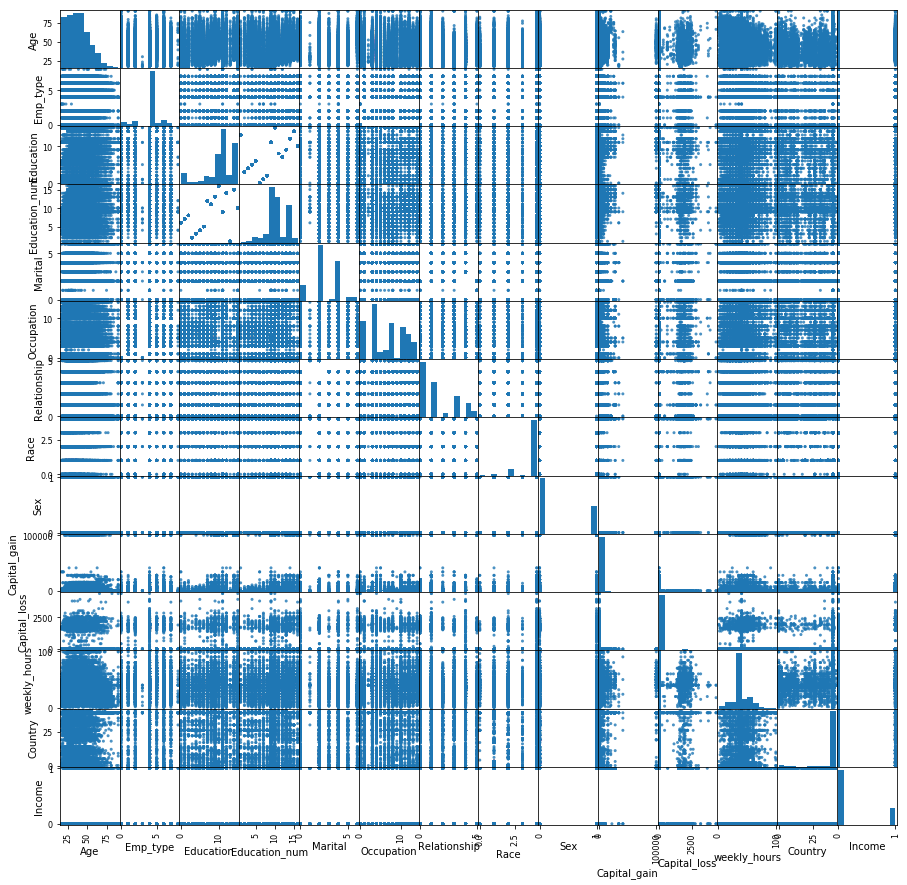
75% 0.000000 45.000000 39.000000 0.000000

max 4356.000000 99.000000 41.000000 1.000000

S-Boxes also show lot of outlier points for most of variables (with exception of Age and Weekly hours). Histograms show that Age is skewed right, Weekly hours has a spike at 40 hours range.







I selected Decision Tree Classifier algorithm because I need an algorithm that works with categorical data (encoded into numbers), and output of the algorithm also categorical (less 50 or more 50). I run algorithm and got weighted average as 0.82.

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False,

random\_state=None, splitter='best')

precision recall f1-score support

0 0.87 0.89 0.88 9243

1 0.63 0.60 0.62 2968

accuracy 0.82 12211

macro avg 0.75 0.75 0.75 12211

weighted avg **0.82** 0.82 0.82 12211

Accuracy level for the model is 81.836%

K-Fold validation algorithm gave me squared error as 0.820 and standard deviation as 0.007

For prediction I created two new data points:

In [20]:

*#Predict new values*

​ model.predict([[39,1,0,0,1,1,1,0,0,10000,500,60,0]])

Out[20]:

array([1])

In [21]:

model.predict([[40,1,0,0,1,1,1,0,0,1000,50,80,0]])

Out[21]:

array([1])

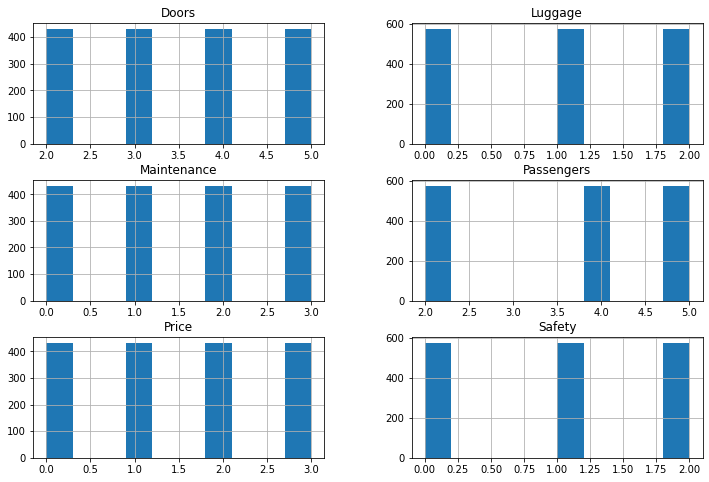
Both sets predicted that salary would be more than $50000.

## PART VI: Machine Learning: Unsupervised

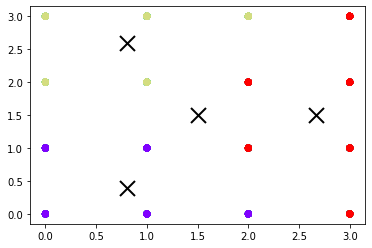
Dataset name is cars\_eveluation.cvs. It has 1728 records/rows grouped by 7 variables. All variables are categorical values. To use in ML algorithm we have to replace categorical values with numerical.

df[[**'Doors'**]]=df[[**'Doors'**]].replace(**"5more"**,5)  
df[[**'Passengers'**]]=df[[**'Passengers'**]].replace(**"more"**,5)  
  
df[**"Doors"**] = pd.to\_numeric(df[**"Doors"**])  
df[**"Passengers"**] = pd.to\_numeric(df[**"Passengers"**])  
Price = Series([0,1,2,3],index=[**'low'**,**'med'**,**'high'**,**'vhigh'**])   
df[**'Price'**]=df.Price.map(Price)  
  
Maintenance = Series([0,1,2,3],index=[**'low'**,**'med'**,**'high'**,**'vhigh'**])   
df[**'Maintenance'**]=df.Maintenance.map(Maintenance)  
  
Luggage = Series([0,1,2],index=[**'small'**,**'med'**,**'big'**])   
df[**'Luggage'**]=df.Luggage.map(Luggage)  
  
Safety = Series([0,1,2],index=[**'low'**,**'med'**,**'high'**])   
df[**'Safety'**]=df.Safety.map(Safety)

Because initial set was categorical, EDA (Exploratory Data Analysis) provides less value. Descriptive analysis shows standard deviation error is very high. Histograms shows equal distribution of data point between categories.



I selected K-Means clustering algorithm which is general purpose algorithm. I selected 4 clusters. But because data points are categorical (i.e. 0, 1, 2, etc.), visualization of clusters does not provide significant value.



For prediction I created two new records:

1. Sports car: very high cost (3), very high maintenance cost (3), with 2 doors, 2 passengers, low luggage (0), medium safety (1) – evaluation could be not good.
2. Another car is 4 door sedan: medium cost (1), high maintenance cost (2), 4 doors, 5 passengers, big luggage (2), high safety (2) – evaluated as good

model.predict([[3,3,2,2,0,1]])

array([1], dtype=int32)

model.predict([[1,2,4,5,2,2]])

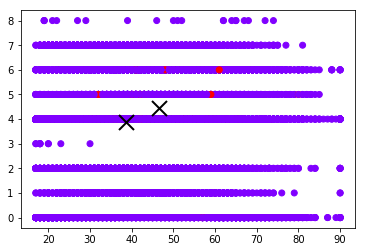
array([2], dtype=int32)

Predictions show that for the 1st set prediction is “acc” and for the 2nd set prediction is “good”.

## PART VII: Evaluate and Compare Machine Learning Models

To compare different ML learning algorithms I used adult\_salary.csv dataset. It has lot of records so it is good to show how different algorithms would work. I used Decision Tree Classifier for Supervised learning and K-Means clustering algorithm for Unsupervised learning. Initial steps to clean the dataset and encode values are the same as within part 5 of the assignment.

K-Means clustering used 2 clusters to separate outcome (one cluster for less than $50k, another for more than $50k salaries).



I used same records to predict outcome for both algorithms. Model1 – supervised ML, Model2 – Unsupervised ML

In [56]:

*#prediction: set 1*

model1.predict([[39,1,0,0,1,1,1,0,0,10000,500,60,0]])

Out[56]:

array([0])

In [57]:

model2.predict([[39,1,0,0,1,1,1,0,0,10000,500,60,0]])

Out[57]:

array([0], dtype=int32)

In [58]:

*#prediction: set 2*

model1.predict([[40,1,0,0,1,1,1,0,0,1000,50,80,0]])

Out[58]:

array([1])

In [59]:

model2.predict([[40,1,0,0,1,1,1,0,0,1000,50,80,0]])

Out[59]:

array([0], dtype=int32)

Because different models trained with different methods prediction for Supervised and Unsupervised algorithms are different. For set 1 both models gave same result “0” (less than $50k). For set 2 Supervised model predicted outcome as “1” (more than $50k), while Unsupervised gave result “0” (less than $50k).

## PART VIII: Final Presentation Videos: YouTube Links