University College of Northern Denmark

Technology and Business

Computer Science Academy Profession (AP) Degree

Class: DMAJ0916 (Big Data)

Title: WASP Report– 4th Semester 2018

Project participants (Group 2):

-Ralfs Zangis

-Andrei-Eugen Birta

Supervisors:

-Nadeem Iftikhar

Due date: 2018-May-24 (14:00 UTC+1)

Submission date: 2018-May-23

Birta *Zangis*

1. Introduction

This document summarizes the collaboration of Group number 2, for the Big Data specialization exam of the 4th Semester. The group consists of two members of two different nationalities. Because of the relatively small size of the group and because of the professional history of the group members, working together; the need for a working contract has significantly decreased and we managed to harness the benefits of diverse ideas and identify multiple possible approaches to certain problems, be it project or working regulations related.

2. Problem Statement

a. Introduction

The purpose of this project is finding a way of reducing or down right preventing all collisions between, both civilian and military aircrafts, and wildlife animals, mainly birds, in the United States of America; by analyzing previous records of such events, and applying various Big Data analyzing techniques to the records.

b. Case Description

Day to day activities thought us that, the collision between something massive and something small and frail, usually ends up pretty bad for the small object and affects little to not at all the massive object. But when it comes to aircraft collisions, with birds and other wildlife creatures, things tend to go bad for both parties. Usually killing the animal and ruining the aircraft, possible for the rest of its “life”.

The following, are images of possible damage that such a collision can cause, to an aircraft.

All 19 passengers and crew died when a Nepalese-based Dornier 228-200 struck a vulture and crashed shortly after take-off in Kathmandu, Sep 2012.

And considering that wildlife population is fluctuating depending on different seasons of the year, but usually increasing in numbers, such collisions should be taken with all seriousness and evaded as much as possible.

According to Allan and Orosz (2001) bird strikes cost commercial air carriers over US$1.2 billion worldwide in the year between 1999–2000. Making the case, not only a safety issue, but also an economical one.

c. Learning Goals

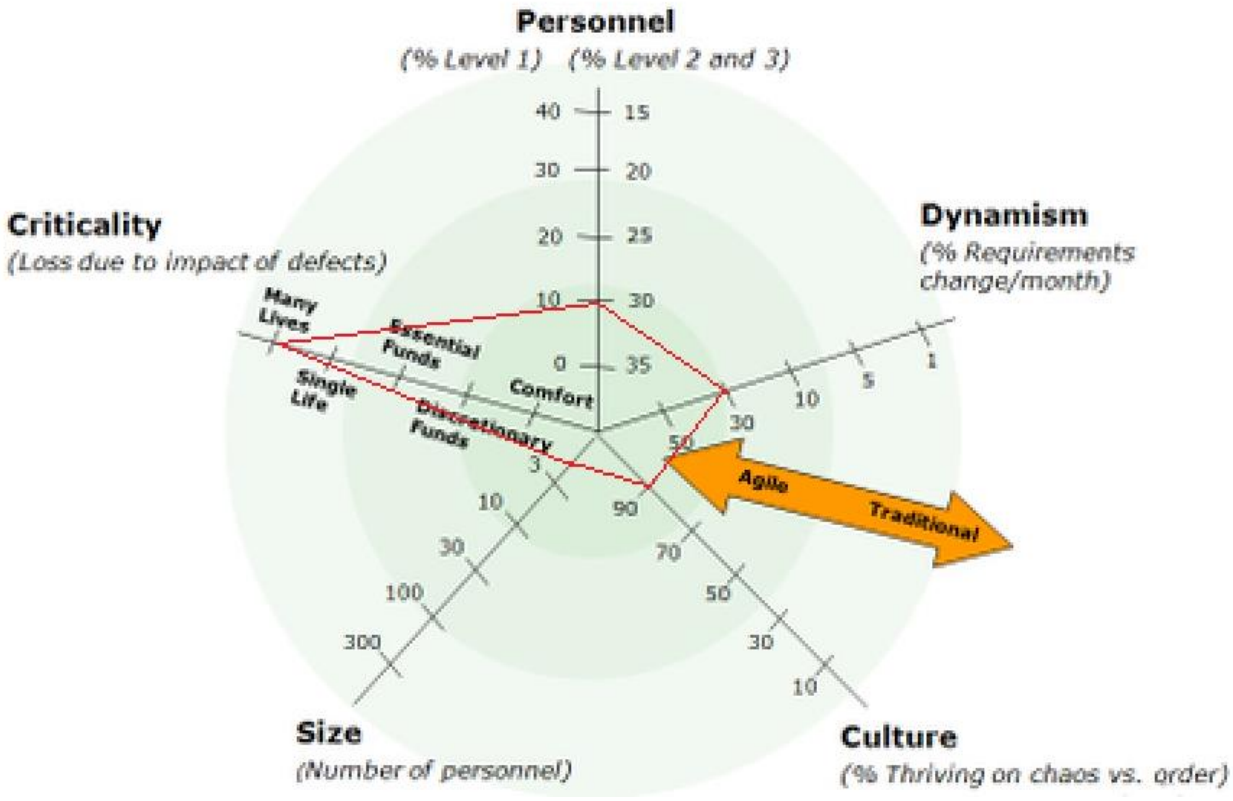
Some of the learning goals for this project are:

- Managing to gather useful datasets related to the case (Data acquisition)
- Converting datasets to a common format, in order to facilitate data analysis, using tools offered by Python
- Wrangling data (dealing with missing values, misspelled words or wrong datatypes), using tools offered by Python
- Describing what and why it has happened, using descriptive and diagnostic analysis techniques
- Predicting whether a future flight is going to partake in such an accident
- Prescribing proven ways of countering or reducing the likelihood of such events happening.

3. Development Framework

Following previously acquired knowledge, from the 3rd Semester's System Development course, we decided that the best way of choosing a development method is by evaluating the team and creating: a Boehm and Turner Model.

The following image is the diagram we have come up with, following the self-evaluation process.



From the diagram above, resulted that we needed some kind of agile development method, due to the high amount of expected changes, small team size, and team's culture, but is structured enough to accommodate for the project's criticality.

Although the high level of criticality indicates that structured development method should be used, we have decided to work, following the Kanban development method, because of two reasons: it best fits the other four measurements and the likelihood of this project being used anything beyond examination reasons, is quite small.

a. Pros

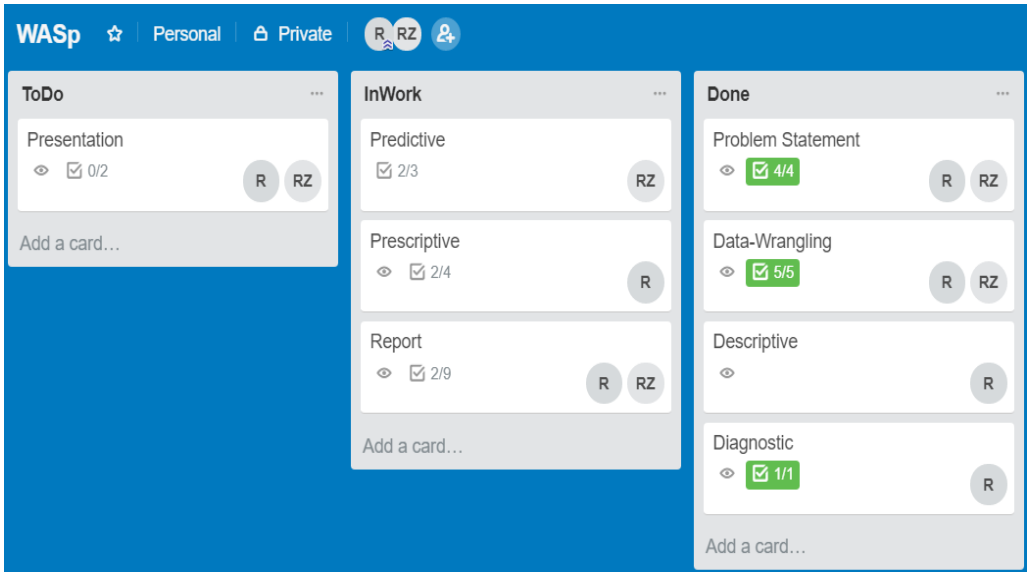
- The small number of guidelines gives the programmer the freedom to work as he pleases;
- The Kanban board helps to manage the project responsibilities and keep track of what has been achieved so far;
- The Kanban board allows for a better understanding of work and workflow.

b. Cons

- Lack of "urgency" concept in the Kanban board, can lead to unnecessary waste of time, due to dependencies of certain tasks, on other tasks.

c. Organizing

Following Kanban's philosophy, we organized our work using a Kanban board. Following image shows our Kanban board, in the state at which it was, during the time this report was constructed:



As can be seen, we divided the board in three main parts:

- ToDo – which holds the tasks that are going to be done in the future;
- InWork – which holds the tasks that are currently worked upon;
- Done – which holds the tasks that have already been finished.

Although Kanban does not fondle with the concept of "urgency", because of the team size and a good grasp over the project's goals and tasks, there was little to no time wasted in waiting for other tasks to be finished.

4. Development Process

Typically, the development process for a Big Data project, depending on the project's needs, starts from one or more small and clearly defined questions, followed by Data Acquisition, Data Wrangling, Descriptive Analysis, Diagnostic Analysis, Predictive Analysis and ending with Prescriptive Analysis; all of them bringing important additions to the overall meaning of the project and helping those who are concerned about the matter, to better understand the situation and take actions based on facts not on feelings.

a. Data Acquisition

Data Acquisition is the first step that has to be made when working on a Big Data related project. This step refers to acquiring the necessary data for answering the previously defined case.

Our datasets were acquired from trusted websites that hold thousands and thousands of various datasets. The exact links for those datasets can be seen in the "References" part of this report.

The datasets we found and acquired are[1]:

- USA Collisions from 1990-1999
- USA Collisions from 2000-2009
- USA Collisions from 2010-2018
- USA Military Collisions from 1990-2018
- USA flights in 2015
- USA airports
- USA airlines

Some datasets were acquired for direct purposes; for example: collision datasets, which have been acquired in order to facilitate Descriptive and Predictive analysis, where other datasets, like airports and airlines, have been acquired in order to better understand the current data and to help in the Data Wrangling process.

b. Data Wrangling

Second step in any Big Data related project, is Data-Wrangling or Data-Cleaning or Data-Cleansing. Although it is referred to under different names, they all denote the same actions, that being: cleaning and curing the data in such ways that it will be ready to go, for further processing.

We chose to do this, by using very powerful libraries, such as "Pandas" and "Numpy", available for Python. Those libraries allow us to process the datasets in a much faster and reliable way than by doing it either manually or using other programming languages such as C# or Java.

[1] each of the following is a different .csv file

The collision datasets were loaded into multiple data frames, using Pandas's "read_csv" function, then merged into one single data frame, just as the following image shows (Figure 1).

```
df1990_1999 = pd.read_csv(location+'STRIKE_REPORTS (1990-1999).csv',
                  encoding = encoding,
                  dtype=types)
df2000_2009 = pd.read_csv(location+'STRIKE_REPORTS (2000-2009).csv',
                  encoding = encoding,
                  dtype=types)
df2010_Current = pd.read_csv(location+'STRIKE_REPORTS (2010-Current).csv',
                  encoding = encoding,
                  dtype=types)
dfMilitary = pd.read_csv(location+'STRIKE_REPORTS_BASH (1990-Current).csv',
                  encoding = encoding,
                  dtype=types)
#putting it into 1 single file
frames = [df1990_1999, df2000_2009, df2010_Current, dfMilitary]
df = pd.concat(frames)
```

Figure 1

b.1. Identify and Handle Missing Values:

Missing values can mess with both the process and the conclusions that further processing of data will result in, thus handling them, is an important part that must be done as quickly as possible, in the development process.

But to be able to handle missing values, first we needed to identify them from among the others. Then replace them with the standard NAN values. Following figure (Figure 2) shows how we managed to achieve that, using Pandas's ".replace" function. Figure 2, also shows us in what form, those values were stored, forms like: "UNKONW", "CHANGE CODE" or "ZZZZ".

```
def fillWithNa(df):
    df = df.replace({'' : np.nan, 'UNKNOWN': np.nan, '[Uu]nknown': np.nan,
                     'UNK': np.nan, 'CHANGE CODE': np.nan, 'ZZZZ': np.nan}, regex=True)
    df['OPID'] = df['OPID'].replace('B-717IT', np.nan, regex=True)

    return df
```

Figure 2

We handled the, now filled in, missing values in different ways, depending on which technique would best fit the situation.

For example, we decided to go with a positive approach and fill N/A values in "nr_injuries" and "nr_fatalities" with 0, going with the presumption that if the incident would've had any casualties, someone, be it a reporter or staff member, would've

looked and made sure that the data is recorded properly. Figure 3, will show exactly how we did that.

```
df[['NR_INJURIES', 'NR_FATALITIES']] = df[['NR_INJURIES', 'NR_FATALITIES']].fillna(0)
```

Figure 3

In some cases, we filled N/A values with the most frequent value, using ".mode", because diagrams, like Figure 4, showed us that the data, in those columns was mostly composed of same value, thus filling the missing values using ".mode" (as can be seen in Figure 5) would result in a much smaller negative impact, in future analysis.

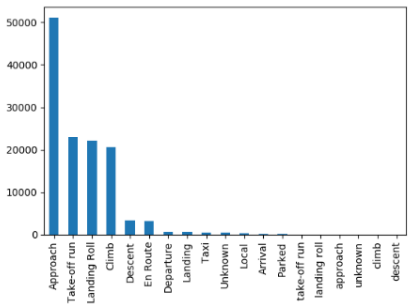


Figure 4

```
df['AC_MASS'].fillna(df['AC_MASS'].mode()[0], inplace=True)
df['NUM_ENGS'].fillna(df['NUM_ENGS'].mode()[0], inplace=True)
        #Object/categorical values
df['SIZE'].fillna(df['SIZE'].mode()[0], inplace=True)
df['BIRDS_STRUCK'].fillna(df['BIRDS_STRUCK'].mode()[0], inplace=True)
df['SPECIES'].fillna(df['SPECIES'].mode()[0], inplace=True)
df['SKY'].fillna(df['SKY'].mode()[0], inplace=True)
df['EFFECT'].fillna(df['EFFECT'].mode()[0], inplace=True)
df['DAMAGE'].fillna(df['DAMAGE'].mode()[0], inplace=True)
df['AC_CLASS'].fillna(df['AC_CLASS'].mode()[0], inplace=True)
df['TYPE_ENG'].fillna(df['TYPE_ENG'].mode()[0], inplace=True)
df['PHASE_OF_FLT'].fillna(df['PHASE_OF_FLT'].mode()[0], inplace=True)
```

Figure 5

In other cases, where we didn't want the outliers to have a great impact on the analysis, we used ".median"(Figure 6). This decision allowed us to better fill in NAN values, especially in places where there are a lot of "0" values, like in columns shown in the following image.

```
df['HEIGHT'].fillna(df['HEIGHT'].median(), inplace=True)
df['DISTANCE'].fillna(df['DISTANCE'].median(), inplace=True)
df['AOS'].fillna(df['AOS'].median(), inplace=True)
```

Figure 6

Columns, that store information like: precipitation, aircraft type or costs, required a much more complex approach, as they are more correlated with other columns. And to keep this correlation, relatively, the same, when filling in NAN values, we decided to fill them by groups.

"COSTS" column was one of the ones affected, in the previously mentioned way, as we noticed that depending on the value in "INDICATED_DAMAGE", the cost also changed, so we created simple method to fill in NAN values based on median in each group (Figure 7).

```
df = fillNaNBasedOnGroup('INDICATED_DAMAGE', 'COSTS', df, 'median')

def fillNaNBasedOnGroup(groupedBy, column, df, strategy):#fills nan values base on theyr group, but doesnt support mode
    df[column] = df[column].fillna(df.groupby(groupedBy)[column].transform(strategy))
    return df
```

Figure 7

Figure 8 shows, a similar approach to the previously mentioned method, except instead of it being used to fill NAN values using a “median” or “mean” approach, it’s being used to fill NAN values, using a “mode” approach.

First, the “helping” column was divided into several groups, then each group was examined and the value that appeared the most was used to fill in the unknown.

```
df = fillNaNwithGroupsMode(df, 'SKY', 'PRECIP')
df = fillNaNwithGroupsMode(df, 'AC_CLASS', 'ATYPE')

def fillNaNwithGroupsMode(df, groupedBy, filled):#fills nan values base on theyr group, supports only mode
    def top_value_count(x):
        return x.value_counts()

    gb = df[[groupedBy, filled]].groupby([groupedBy])[filled]
    df_top_freq = gb.apply(top_value_count).reset_index()
    df_top_freq = df_top_freq.sort_values(by=filled, ascending=False)
    df_top_freq = df_top_freq.drop_duplicates([groupedBy], keep='first')

    for i in df_top_freq.iterrows():
        df.loc[df[groupedBy] == i[1][0], filled] = df.loc[df[groupedBy] == i[1][0], filled].fillna(i[1][1])

    return df
```

Figure 8

Some columns had to be dropped all together, simply because they were consisting of mostly NAN values and would bring little to no benefits in the next phases. And removing them would bring us, increased performance related benefits. Or because they contained duplicate data, like “Incident_month” and “Incident_year”, data, which is already stored in “Incident_date”.

b.2. Data Formatting:

The purpose of Data-Formatting is to make sure that each column in the data frame, is of right type. This would help improve all the further processing done to the data frame, both from a performance point of view and a data-quality point of view.

Since we loaded the datasets into data frames, using special column types (Figure 9), there was little to no additional work needed in this part, except when dealing with datetime values.

```python
def loadFiles():
    location = 'Extracted_dataset/'
    encoding = 'ISO-8859-1'
    types = {'INDEX_NR': int, 'OPID': object, 'OPERATOR': object, 'ATYPE': object, 'AMA': object,
                            'AMO': object, 'EMA': object, 'EMO': object, 'AC_CLASS': object, 'AC_MASS': float,
                            'NUM_ENGS': float, 'TYPE_ENG': object, 'ENG_1_POS': object, 'ENG_2_POS': float, 'ENG_3_POS': object,
                            'ENG_4_POS': float, 'REG': object, 'FLT': object, 'REMAINS_COLLECTED': bool, 'REMAINS_SENT': bool,
                            'INCIDENT_DATE': object, 'INCIDENT_MONTH': float, 'INCIDENT_YEAR': float, 'TIME_OF_DAY': object, 'TIME': float,
                            'AIRPORT_ID': object, 'AIRPORT': object, 'STATE': object, 'FAAREGION': object, 'ENROUTE': object,
                            'RUNWAY': object, 'LOCATION': object, 'HEIGHT': float, 'SPEED': float, 'DISTANCE': float,
                            'PHASE_OF_FLT': object, 'DAMAGE': object, 'STR_RAD': bool, 'DAM_RAD': bool, 'STR_WINDSHLD': bool,
                            'DAM_WINDSHLD': bool, 'STR_NOSE': bool, 'DAM_NOSE': bool, 'STR_ENG1': bool, 'DAM_ENG1': bool,
                            'STR_ENG2': bool, 'DAM_ENG2': bool, 'STR_ENG3': bool, 'DAM_ENG3': bool, 'STR_ENG4': bool,
                            'DAM_ENG4': bool, 'INGESTED': bool, 'STR_PROP': bool, 'DAM_PROP': bool, 'STR_WING_ROT': bool,
                            'DAM_WING_ROT': bool, 'STR_FUSE': bool, 'DAM_FUSE': bool, 'STR_LG': bool, 'DAM_LG': bool,
                            'STR_TAIL': bool, 'DAM_TAIL': bool, 'STR_LGHTS': bool, 'DAM_LGHTS': bool, 'STR_OTHER': bool,
                            'DAM_OTHER': bool, 'OTHER_SPECIFY': object, 'EFFECT': object, 'EFFECT_OTHER': object, 'SKY': object,
                            'PRECIP': object, 'SPECIES_ID': object, 'SPECIES': object, 'BIRDS_SEEN': object, 'BIRDS_STRUCK': object,
                            'SIZE': object, 'WARNED': object, 'COMMENTS': object, 'REMARKS': object, 'AOS': float,
                            'COST_REPAIRS': float, 'COST_OTHER': float, 'COST_REPAIRS_INFL_ADJ': float, 'COST_OTHER_INFL_ADJ': float, 'REPORTED_NAME': object
                            'REPORTED_TITLE': object, 'REPORTED_DATE': object, 'SOURCE': object, 'PERSON': object, 'NR_INJURIES': float,
                            'NR_FATALITIES': float, 'LUPDATE': object, 'TRANSFER': bool, 'INDICATED_DAMAGE': bool}

    df1990_1999 = pd.read_csv(location+'STRIKE_REPORTS (1990-1999).csv',
                      encoding = encoding,
                      dtype=types)


    df2000_2009 = pd.read_csv(location+'STRIKE_REPORTS (2000-2009).csv',
                      encoding = encoding,
                      dtype=types)

    df2010_Current = pd.read_csv(location+'STRIKE_REPORTS (2010-Current).csv',
                      encoding = encoding,
                      dtype=types)

    dfMilitary = pd.read_csv(location+'STRIKE_REPORTS_BASH (1990-Current).csv',
                      encoding = encoding,
                      dtype=types)

    #putting it into 1 single file
    frames = [df1990_1999, df2000_2009, df2010_Current, dfMilitary]
    df = pd.concat(frames)

    return df
```

Figure 9

b.3. Data Normalization:

Data-Normalization refers to making sure that all data is within the same range, to make sure that there are no misspells or wrong values in the wrong places.

One of the more important parts in this section, was making sure all the date-time stamps are saved in same format and in same field (as opposed to having 1 field for the date and one for the time). We achieved this using Pandas's "to_datetime" function. Another problem which we identified here was: sometimes the time was recorded with "dusk" or "dawn" or other words that can describe the time of day, instead of an actual hour. This was dealt with by replacing those values with the most common hour for each time of day, described by the specific word.

Another important part of this section, was making sure the airport names are not misspelled or wrongly typed in any way, shape of form, which we achieved by checking all the values within the column and ".replace" the wrong values with the appropriate ones using the USA airport dataset.

b.4. Indicator variables or dummy variables

Indicator variables or dummy variables are columns which hold “label” information, for example: “gas-type” is a dummy variable, which tells us the type of fuel that an engine requires to run, where “km-per-liter” is a column that holds “feature” data.

In our case, we have several columns which serve as dummy variables, some of which are: “Type-Eng” (tells the type of engine), “Str_Wing_Rot” (struck wing), “Str_Windshield” (struck windshield) and “Str_Tail”(struck tail) and many more.

c. Descriptive Analysis

Descriptive analysis is the part in a big data project, where the Data scientist takes a look at the archived, cleaned data and makes sense out of it. He analyzes it, creates charts and figures, that would ultimately lead him and all concerned parties, into achieving their final goal. In our case, the goal is trying to reduce or down right stop future collisions between aircrafts and wildlife, and in order to do so, we first need to know how the collisions happen.

c.1. Used Tools

For this part, we decided to use “Power Bi”, a business analytics service provided by Microsoft. We decided to use “Power BI”, because it provides interactive visualizations with self-service business intelligence capabilities and because it’s one of the tools that we were though using, tying together knowledge from multiple courses.

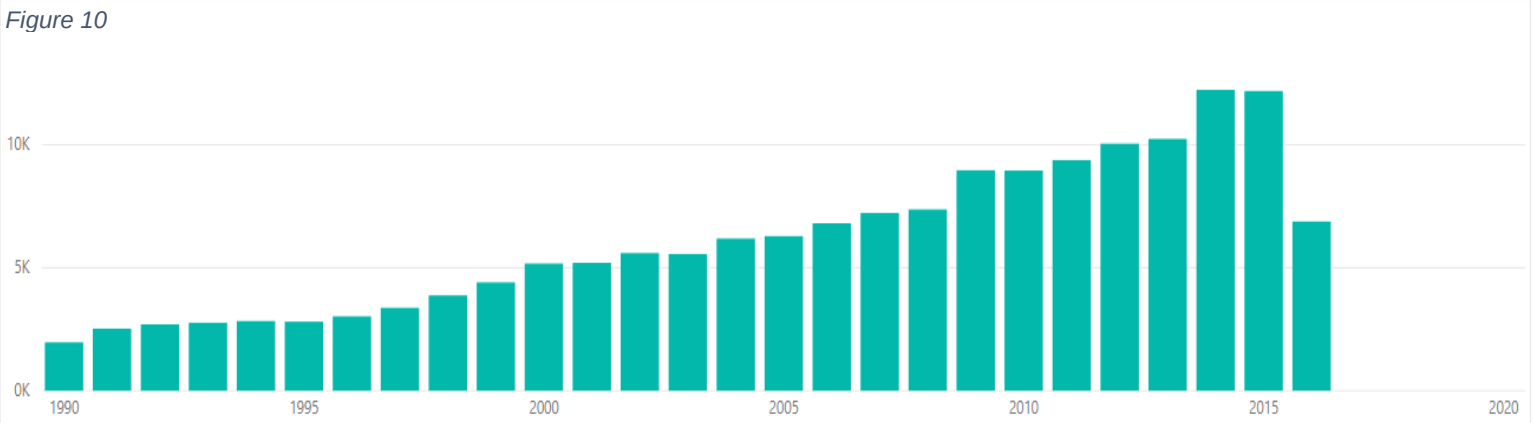
Other tools that we have considered for this part are: Python and Tablueau. Tools which were not chosen because of their pricing and, sometimes, unreliable website (in case of Tableau) and higher grade of difficulty to use (in case of Python).

c.2. Findings

So, what can we see from previous events?

Certain things can be observed from the datasets we have acquired, the following picture(Figure10) shows that, although we’ve seen a major decrease in recent times, this type of accidents is following a growing trend, and every year, more and more collisions happen.

Figure 10



We can also observe a seasonal pattern. Figure 11 is a combination of data from (1990-2016), divided per month, and shows that most accidents happen during mid-late summer and early-mid fall. And although this season has the most accidents, the number of injuries (represented in green), stays relatively the same, fluctuating, relatively little; the number of fatalities (represented in black) is at its lowest value from the entire year. With January and March being deadliest months.

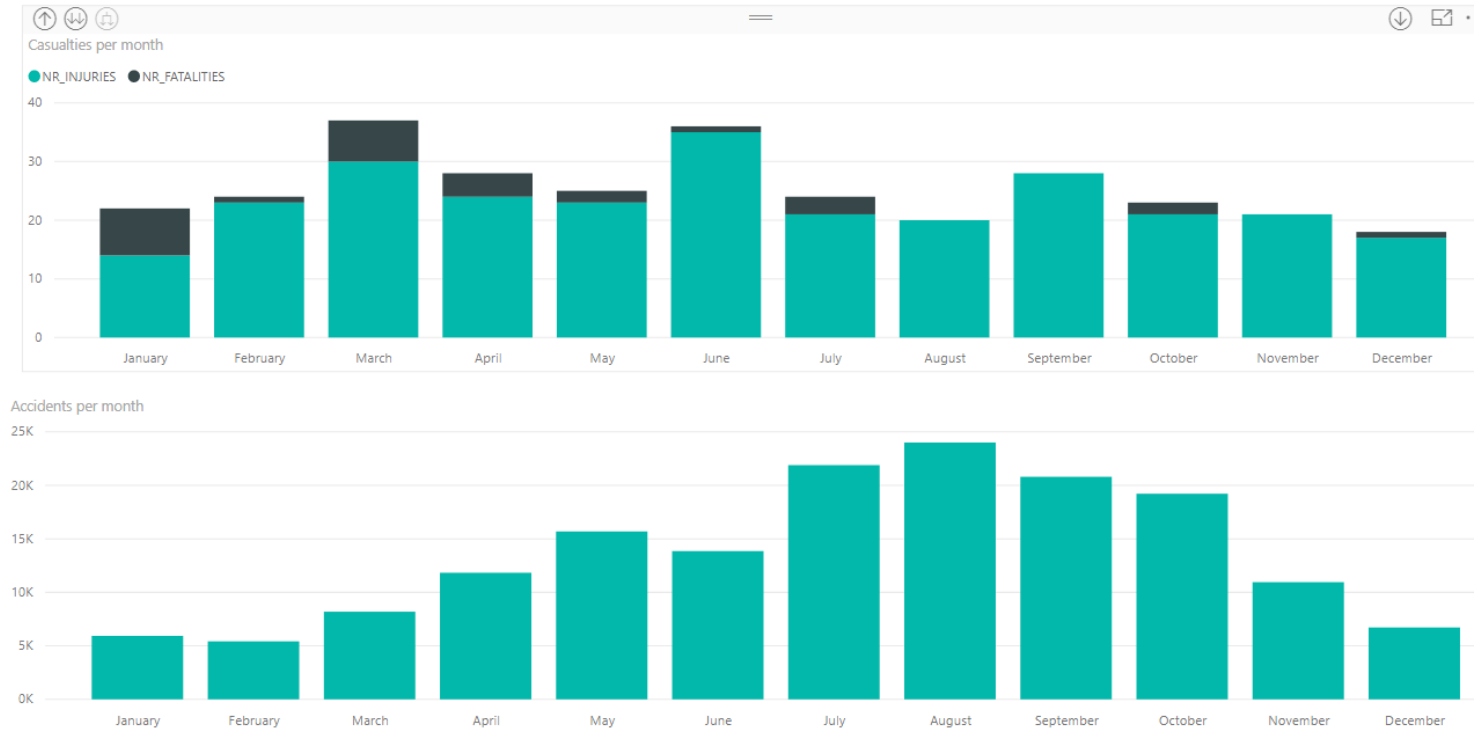


Figure 11

Figure 12 shows us that most (more than 50%) of the collisions happened at 50 feet (~16 meters) or less, meaning that most collisions happened before or during, take-off or landing.

Another conclusion that can be drawn from this, is that 90% of the collisions happen at or in the near vicinity of an airport.

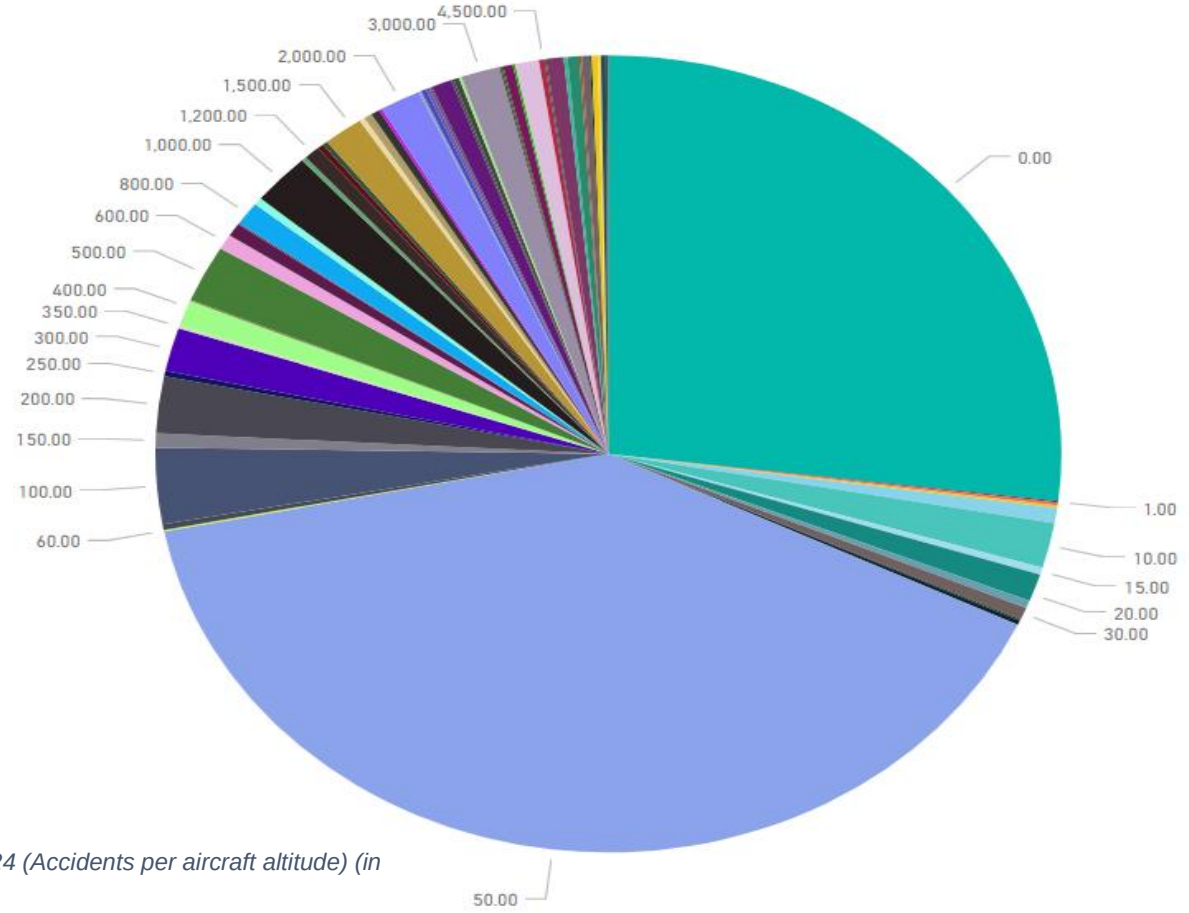


Figure 124 (Accidents per aircraft altitude) (in feet)

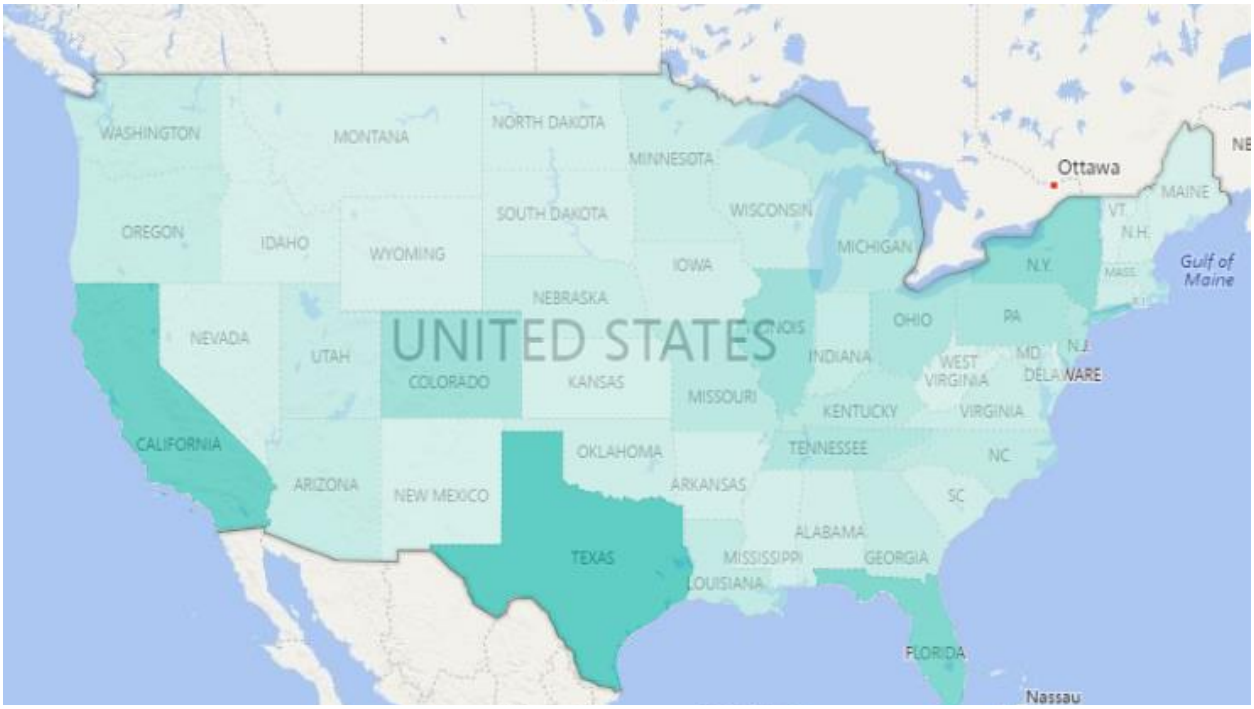


Figure 13

Figure 13, is a heat map, showing which state suffers the most from collisions with wildlife. Comparing to Figure 14 (which shows the busiest states, in terms of flights), we can see that while some states, like Texas and California are expected to have more collisions, due to their increased number of flights, other states, like Nevada and Utah don't seem to be affected by that at all.

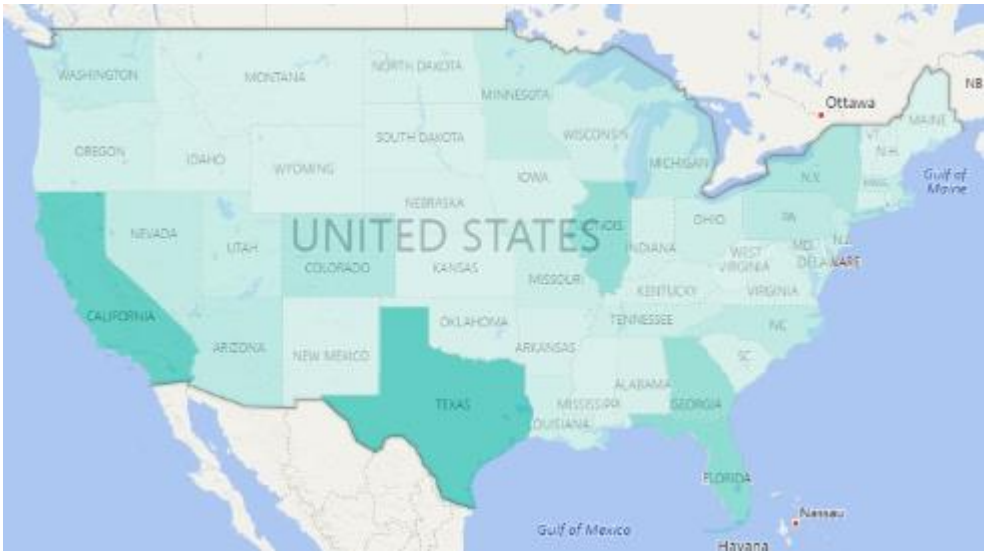


Figure 14 (busiest states by flights)

Taking a deeper look at the previous finding, Figure 15 is a table which represents the busiest airports. While Figure 16 is a table which represents the airports with most collisions.

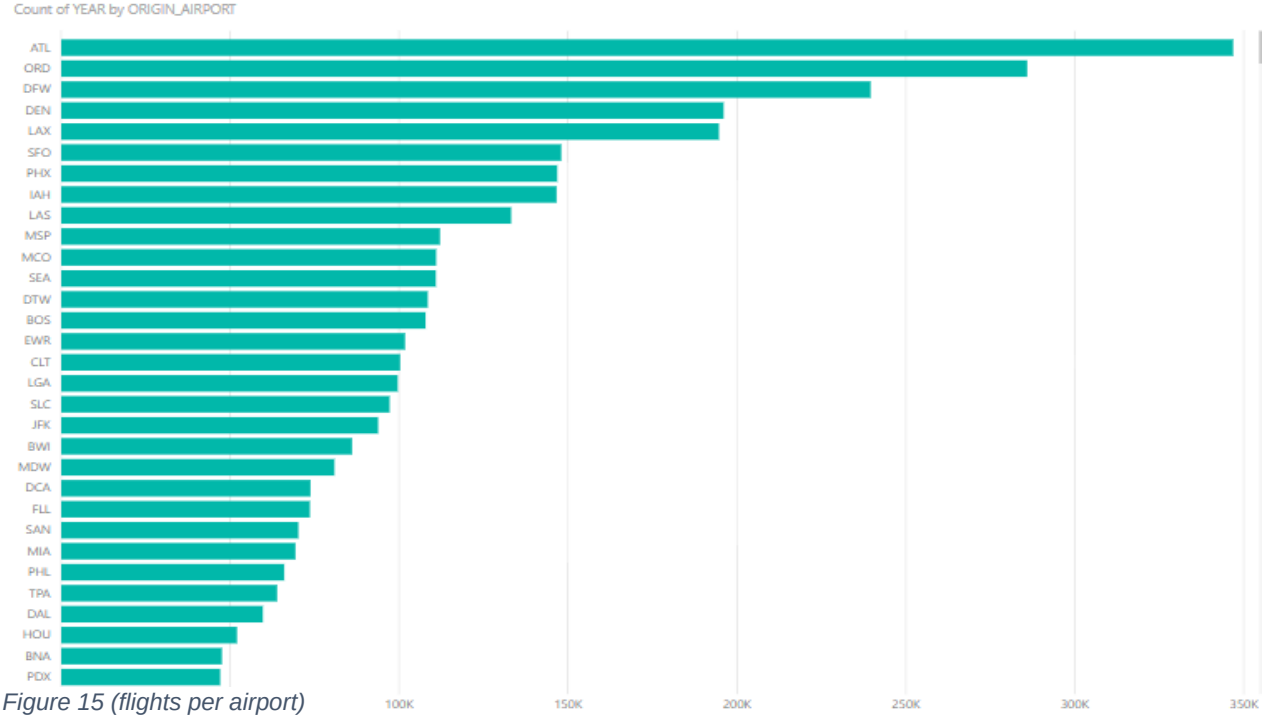


Figure 15 (flights per airport)

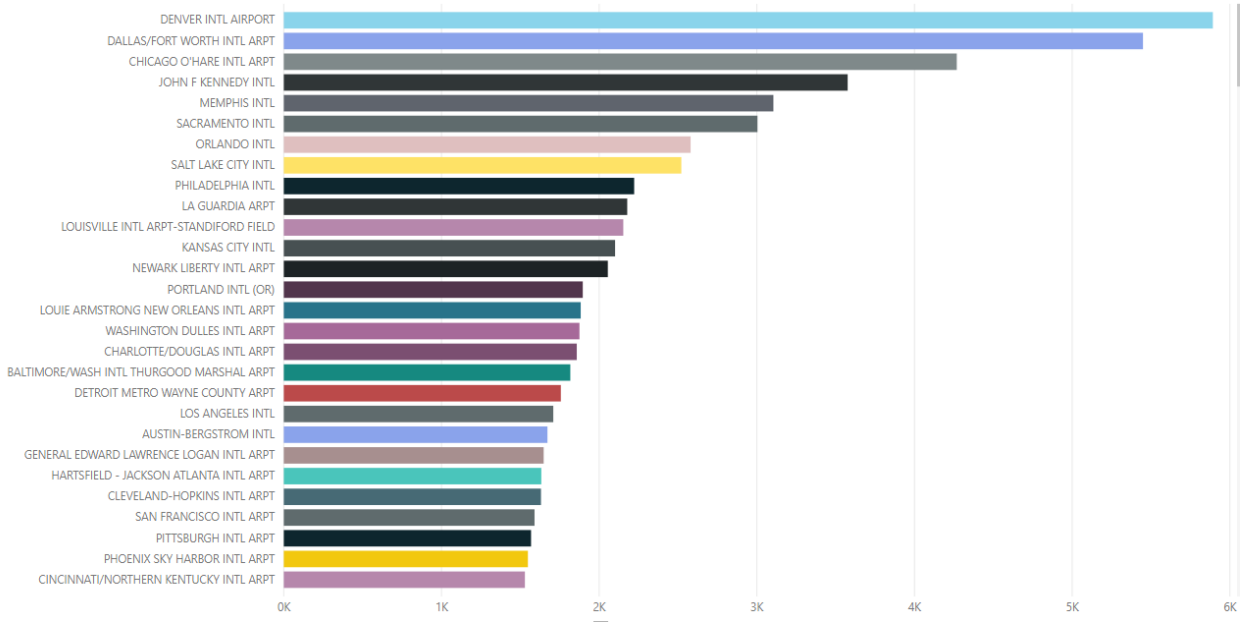


Figure 16 (collisions per airport)

We can see that although the DEN airport, is the 4th busiest airport in USA, it has the most collisions with wildlife; and ATL (Hartsfield–Jackson Atlanta International Airport), the busiest airport in USA, ranks place 23rd on the collisions list, meaning that the number of flights, does not have that much of an impact on the collision rate.

Figure 17 shows which are the most common species, aircrafts collide with.

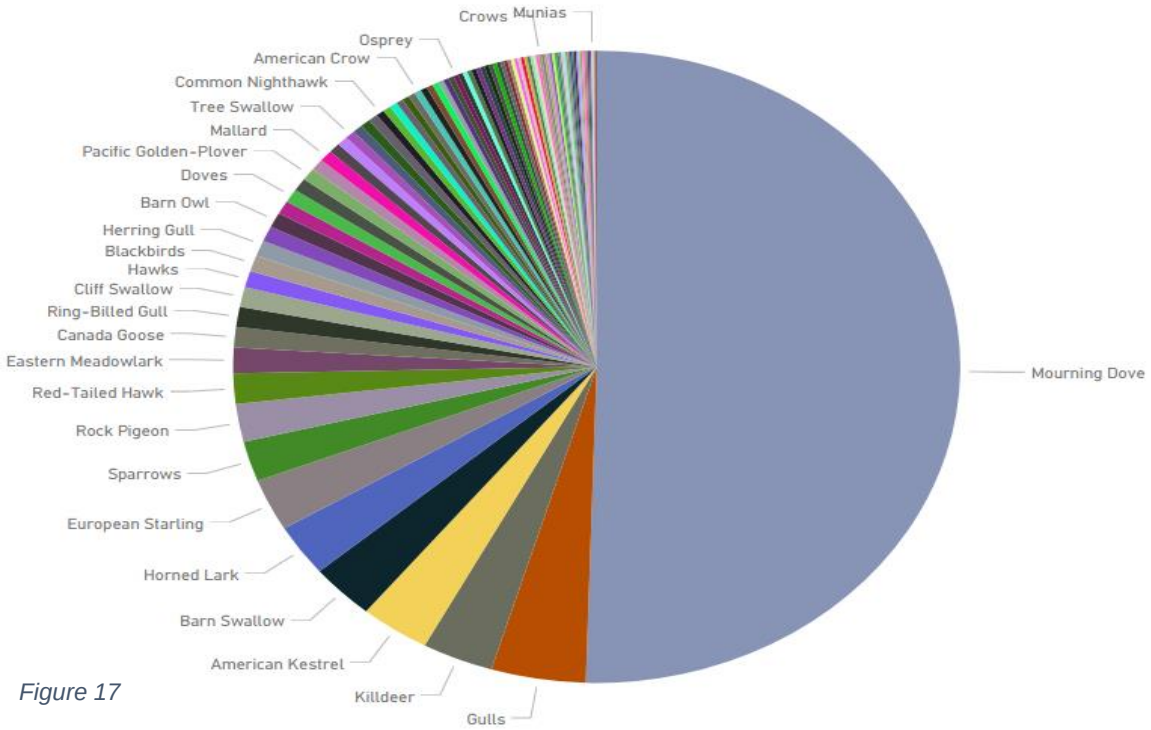


Figure 17

Clearly showing that some species are more likely to partake in a collision, than others.

Figure 18 shows the distribution of bird strikes onto a typical aircraft. 44% of the collisions affect the engine rendering the aircraft useless for a period of time, tightly dependent on the aircraft type and the availability of replacement parts. The wings and the engine are the only parts of an aircraft that hold fuel, and a collision to either of them, can cause the engines to explode, putting in danger the passenger's lives or causing fuel leakages, forcing the aircraft to execute an emergency landing.

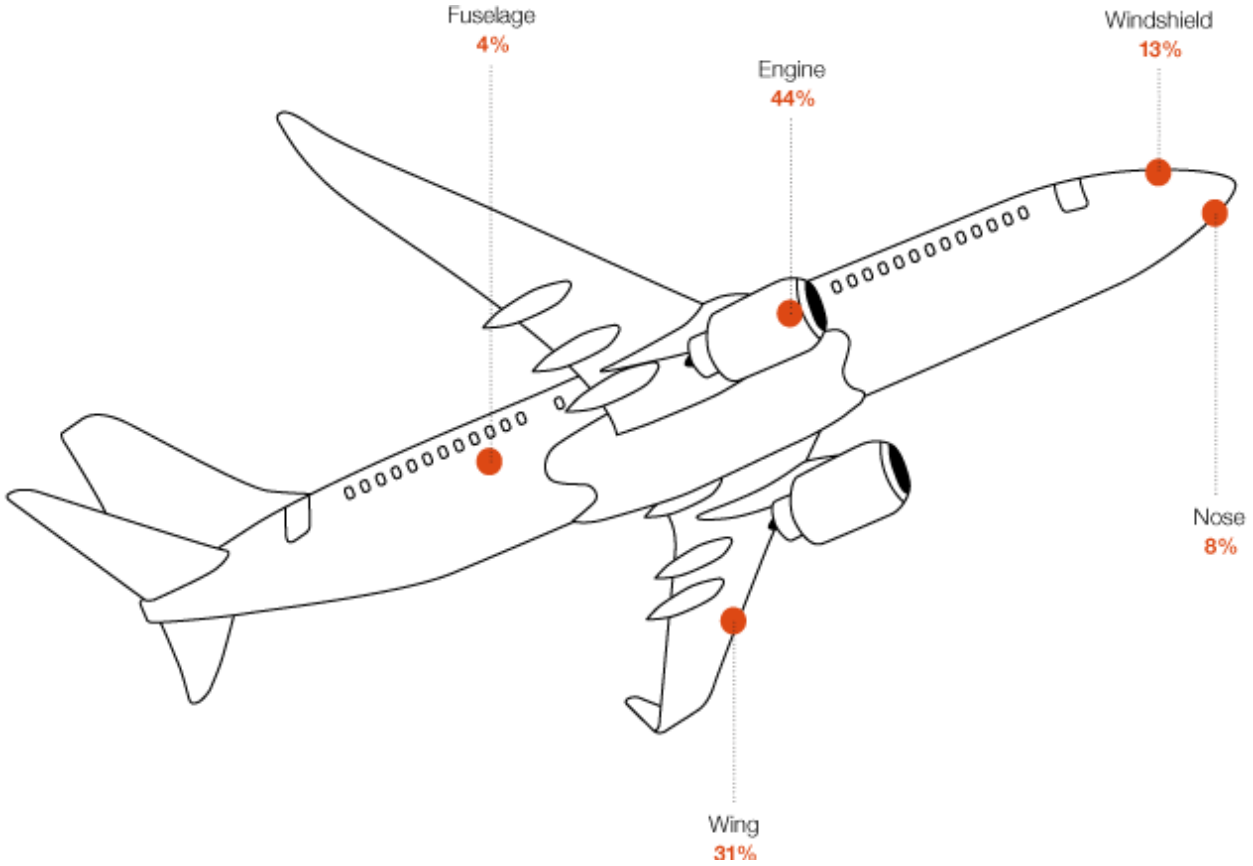


Figure 18

d. Diagnostic Analysis

Diagnostic Analysis is the part in which the Data scientist finds the reasons why did the identified issues, in Descriptive Analysis, happened.

We managed to find answer to the following questions, by researching the matter on the internet, where sources like books, articles and airplane manufacturer websites, have served us a great favor in answering some of these questions[2].

- How come, more than 50% of the collisions happen at altitudes bellow 16 meters?

[2] Said sources, can be found in the "References" part of this report.

- Why do some months have part of more accidents than others?
- How come some of the busiest airports and states are not the same as those that have the most crashes?
- Why are some species more likely to hit an aircraft than others?

In order to answer those questions, we needed more information.

According to a book written by Kaufman Kenn ("Lives of North American Birds", 1996), Mourning Doves, the bird species which was part of 47% of all collisions since the 90's, like to spend their time at altitudes between 5 feet (1.5meters) and 25 feet (7.6 meters), giving the reason why the Mourning Doves are more likely to crash into a plane, than any other species. "Adding salt to the wound" next species with the highest collision rate, are diurnal birds, natural predators to the Mourning Dove, be it for their meat, nests or eggs.

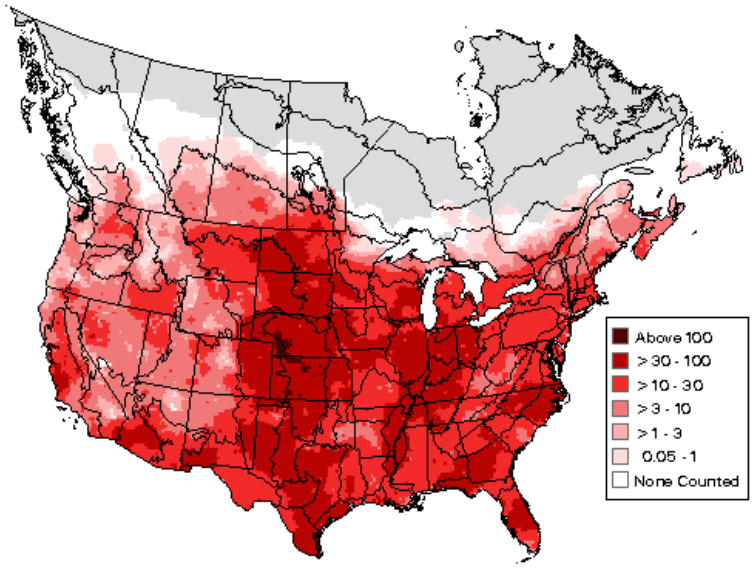


Figure 19 Breeding distribution and relative abundance of the Mourning Dove in North America based on the federal Breeding Bird Survey from 2011 to 2015 (Sauer et al. 2017).

Figure 19 represents a heat map of Mourning Doves estimated population, over the north American territory. Comparing this one with Figure 13, we can see a clear resemblance between the two, showing part of the reason why some of the states have more collisions with Mourning Doves, than others. And although not all of the collisions are with Mourning Doves, next species that are likely to be part of a collision are natural predators to the Mourning Doves, and where there're Doves, there're predators as well, further increasing the risk that a plane will crash into a bird.

Most collisions happen in the summer-fall season as that is the time most birds migrate to warmer territories. January and March have part of more deaths as that is the time when bigger birds dominate the sky. And a bird's body is directly proportional with the damage an aircraft will sustain during an accident.

e. Predictive Analysis

Predictive analysis encompasses a variety of statistical techniques, but we chose to use machine learning to analyze current and historical facts, which would allow us to make predictions about future or otherwise unknown events.

Will it happen again?

In order to predict whether a flight will be part of a collision accident, we used a 2nd dataset which contained all flights in 2015, dataset which had ~6 million entries of flights on USA grounds. After cleaning the dataset and identifying which of those flights were crashes (using the initial dataset, which contained only crashed flights), we proceeded with the predictive analysis.

Our choice regarding this project was between supervised and unsupervised learning, and since this project was partly made for learning purposes, we chose to implement both, in order to see what each of them is good at and get a better understanding on which best fits a certain situation.

e.1. Used tools

For this part, we decided to use various libraries available for the Python language.

Other tools which we considered is: Eclipse Deeplearning4j, a tool used for deep neural networks and is designed to be used in business environments on distributed GPUs and CPUs, which did not fit our project since, as can be seen in “Unsupervised Learning” we decided that neuronal networks are something we cannot implement, given the current skills that we have.

e.2. Supervised Learning

Supervised learning maps an input to an output based on example input-output pairs. It uses labeled data consisting of features (X - input object) and labels (Y - a desired output). Using this method, we managed to create a model that is capable of predicting, whether a flight will partake in a wildlife collision or not, with a quite astonishing accuracy.

There are two types of problems, that supervised learning can solve:

- Classification: A classification problem is when the output variable is a category, such as “red” or “blue”.
- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

e.2.1. Chosen technique

There are multiple ways of doing Supervised Learning, and out of all of those, we decided to use decision tree, since it came out as best among the other techniques which we considered.

Other techniques which we considered are:

- Naïve Bayes but because it is biased towards common results, and since the chance of a plane partaking into a collision with wildlife is very small, we considered that the prediction accuracy would take a hit, were we to use this technique.
- Logistic Regression but because it is used to estimate discrete values based on given set of independent variables and because we wanted a concrete answer (whether the flight will crash into wildlife or not) as opposed to “There is 60% chance the flight will crash”, we decided not to implement this one.

e.2.2. Technique implementation

As shown in Figure 20, we started by separating data frame into features and labels. Then we decided how many rows of this dataset we would like to predict. Using “cross_vaidation” we split this dataset, for training purpose using the following ratios: 70% training and 30% testing. After this we executed the model and got an astonishing 99.53% prediction accuracy without the use of Unsupervised Learning and 99.55% prediction accuracy with the help of Unsupervised Learning.

```
#Supervised learning using the decision tree
#X- feature
#y- lable
def Supervised(df, predict = 100, Column_y = 'CRASHED'):
    X = df.drop([Column_y], axis=1)
    y = df[[Column_y]]

    X_train, X_test, y_train, y_test = cross_validation.train_test_split(X[:-predict], y[:-predict], test_size=0.3, random_state=7)
    model = tree.DecisionTreeClassifier()
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    print(accuracy)

    print(model.predict(X[-predict:]))
    print(y[-predict:])
```

Figure 20

e.3. Unsupervised Learning

Unsupervised learning describes hidden structure of "unlabeled" data. Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the algorithm, it is used to predict pattern in data —which is one way of distinguishing from supervised. There are 3 main types of unsupervised learning techniques:

- Clustering: used to form groups in such a way that all individual entries in a group are similar to one another.
- Anomaly detection: used to identify anomalies in a system. Ex: bank frauds.
- Neural networks: system which is capable of learning. Ex: Image recognition, speech recognition, etc.

e.3.1. Chosen Technique

There are multiple ways of doing Unsupervised Learning, and out of all of those, we decided to use a clustering technique called K-means. The reason we chose to do so, is not only because K-means is one of the, if not the most popular technique, used in situations similar to this one, but also because K-means would help us in improving the Supervised learning model, previously mentioned, by grouping the flights into several clusters, further curing the data and forcing the model to predict based on available data, for similar flights, as opposed to predicting based on all data, indifferent on how related, and possibly, how useful it is. This would ultimately help the model by taking away possible outliers, for the specific flight, and also help the user by reducing the required hardware requirements, as it will force the Supervised model to compare the flight using flights that are part of the same cluster.

Other techniques that we considered are:

- Hierarchical clustering, which is a method of clustering that seeks to group data using, either an Agglomerative(bottom-up) or Divisive(top-down) approach. Technique which we did not choose because, as many results on the internet showed, is slow and requires a lot of memory, and wasn't suitable solution to our limitations.
- Deep belief network, which is a type of neuronal network, that is composed of several layers connected with one another, which is capable of adapting and learning to better predict future outcomes. Technique which we did not choose because it required a deeper level of knowledge and understanding on the matter of neuronal networks, than we currently have, and it doesn't achieve goals set for our project regarding unsupervised learning.

e.3.2. Technique Implementation

We started the work on unsupervised learning, by (as seen in Figure 21) deciding the amount of values, the model, would be tested with. Then we decomposed the data into two components and proceeded to decide number of clusters (done using elbow analysis) and then thought the algorithm using the extracted information. After that, we added the "clusters" column to data frame, so we can store the results of clustering and

use it for other tasks. At the end we continued by displaying the clusters and predicting in which of the four clusters, the new values would fit.

```
#Unsupervised learning using k means
def Unsupervised(df, predict = 100, clusters=4):
    #Scaling- mean=0 std=1
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(df)

    reduced_data = PCA(n_components=2).fit_transform(X_scaled[:-predict])
    kmeans = KMeans(init='k-means++', n_clusters=clusters, n_init=10).fit(reduced_data)
    df['clusters'] = np.nan
    df['clusters'][:-predict] = kmeans.labels_

    #Analyze the clusters
    print(df.groupby(['clusters']).mean())
    unique_elements, counts_elements = np.unique(kmeans.labels_, return_counts=True)
    print(np.asarray((unique_elements, counts_elements)))

    #Visualization of actual data
    plt.scatter(reduced_data[:,0], reduced_data[:,1], c=kmeans.labels_, cmap='rainbow')
    plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], marker='x', color='w')
    plt.title('K-means clustering on the wildlife-collisions dataset')
    plt.xticks(())
    plt.yticks(())
    plt.show()

    #Prediction
    reduced_data = PCA(n_components=2).fit_transform(X_scaled[-predict:])
    Z = kmeans.predict(reduced_data)
    df['clusters'][-predict:] = Z
    unique_elements, counts_elements = np.unique(Z, return_counts=True)
    print('Prediction results:')
    print(np.asarray((unique_elements, counts_elements)))

    return df
```

Figure 21

Figure 22 is a plot, which visually represents the clustered flights, as the model decided to distribute them. Also, in each cluster there is a white cross, which represents the mean value for that cluster. This plot also helped us see that flights in some clusters are more likely to partake in an accident, than others; in fact, the worst cluster is three times more likely to crush with wildlife than the others.

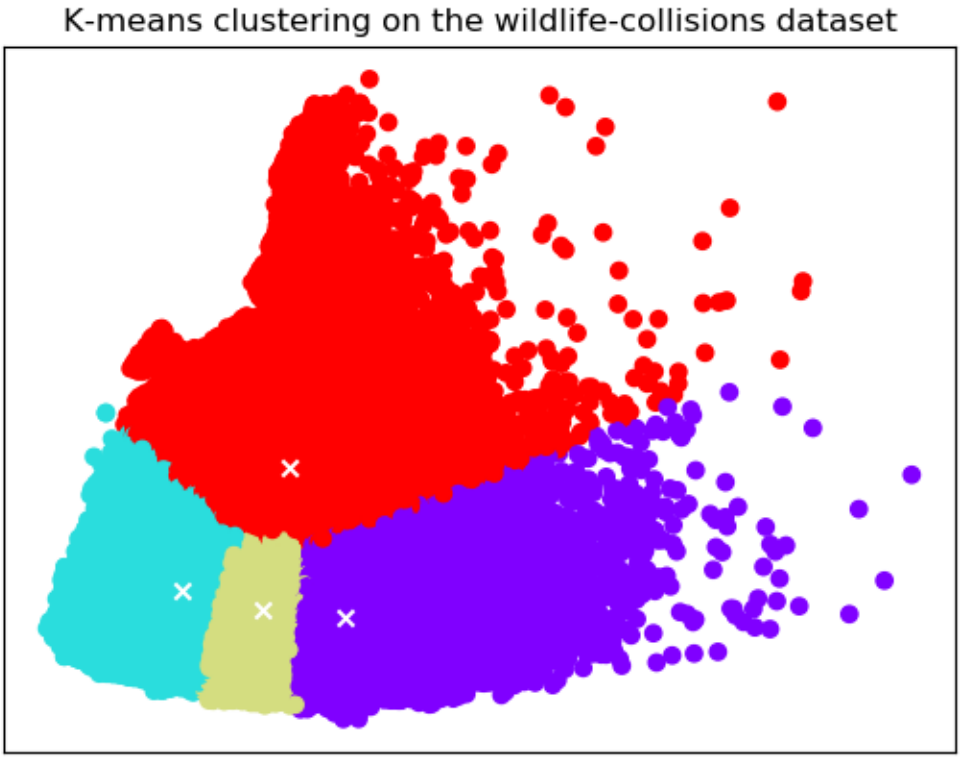


Figure 22

How did you decide to have four clusters?

To decide the number of clusters we had the choice of using either Silhouette Analysis or Elbow Analysis, we chose latter one, as it was both simple to implement and work with. Chosen method looks at the percentage of variance with different number of clusters.

Figure 23 shows how we started working on the Elbow Analysis, by scaling our dataset, so our data isn't affected by outliners, making mean value 0 and standard deviation of 1. Following this step, we executed the algorithm 10 times with different number of clusters and in the end the percentage of variance was plotted (Figure 24) to visualize the results.

```python
#Deciding how many cluster are needed by using the Elbow Analysis
def ElbowAnalysis(df):
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(df)

    cluster_range  = range( 1, 10 )
    cluster_errors = []

    for num_clusters in cluster_range:
        print(num_clusters)
        clusters = KMeans( num_clusters )
        clusters.fit( X_scaled )
        cluster_errors.append( clusters.inertia_ )

    clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors": cluster_errors } )

    print(clusters_df)

    plt.figure(figsize=(12,6))
    plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
    plt.show()
```

Figure 23

Figure 24 shows us the result of plotting the information received by running the method mentioned above. As can be seen, the variance, starts dropping slower at 4th cluster, which lead to our decision of using only four clusters for further processing. Of course, having more than four clusters would possibly further improve the accuracy, but the improvement would be far less than the toll paid performance wise.

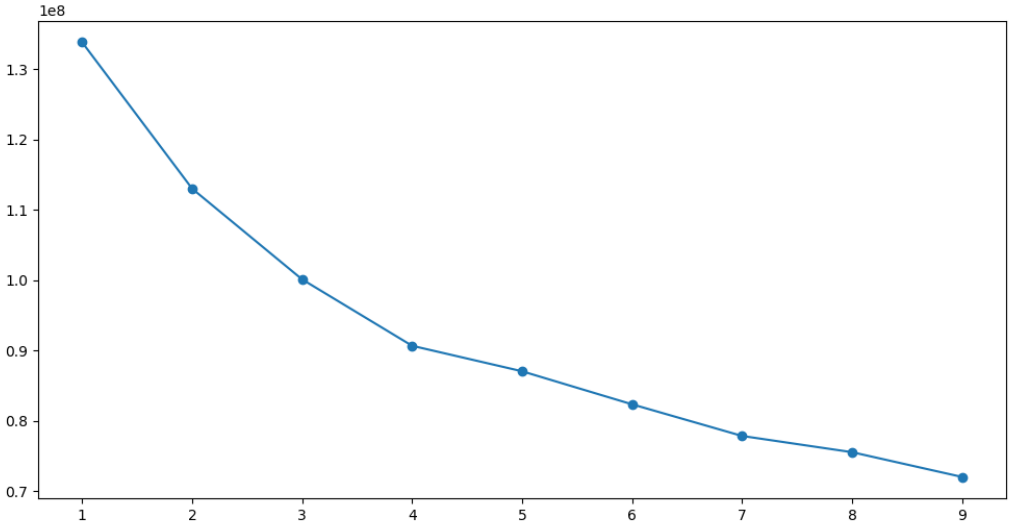


Figure 24

f. Prescriptive Analysis

The Prescriptive analysis is the analysis which gives the concerned authorities, a "prescription", some advice on how to reach the goal or, in this case, how to prevent collisions from happening.

Following on Figure 12's findings, about how much of an impact the airport itself and its location has on a collision. We deduced, that airport is indirectly responsible for around 90% of the collisions, which happen at or nearby airports. Meaning that, perhaps, the best way of reducing collisions is to better manage the airspace near airport.

What can be done, in order to stop collisions from happening?

According to an article by Melissa Mayntz "Airport Bird Control Methods", there are three main ways of minimizing airplane-wildlife collisions, most successful airports have reported using a combination of all three:

- Modifying the Habitat:
  - Removing seed-bearing plants to eliminate food sources
  - Covering nearby water sources with netting to prevent birds from landing
  - Removing brush and trees that serve as attractive nesting sites
  - Keeping grass mowed short so it is not as suitable for bird shelter
- Controlling bird behavior:
  - Using sonic cannons, recorded predator calls and other noise generators to disrupt birds
  - Using lasers at dawn and dusk to simulate predators and scare birds away
  - Flying trained falcons over roosting areas to disrupt birds before they nest
  - Training dogs to track through the habitat and teach birds that the area has many predators
- Controlling aircraft behavior:
  - Training spotters with binoculars and scopes to pinpoint hazardous bird's areas and directing planes to different runways or approaches
  - Using radar equipment to track the movement and density of bird flocks to predict their behavior and manage control techniques more effectively
  - Adjusting flight times to avoid the busiest hours for bird activity, such as early morning and late evening or during peak migration periods

According to Roger Nicholson, Ph.D., Associate Technical Fellow, Aviation System Safety, and William S. Reed, Safety Pilot, Boeing Flight Technical and Safety; some of the best preventive measures against wildlife collisions are:

- Delaying the takeoff or landing when fuel permits, in case any wildlife is spotted
- Takeoff or landing on another runway, where no wildlife is present.
- Thorough preparation and training effectuated by all crew members, be it pilot or flight assistant.

According to Michael Begier, national coordinator of the airport wildlife hazards program at the US Department of Agriculture; the use of pyrotechnics, like flash bangs, gun shots or fireworks, has been proved to be quite efficient in distracting birds and scaring them away from a runway.

Ellen Lindblad director of planning and environmental compliance at Southwest Florida International Airport has employed Sky, a canine helper that managed to reduce the wildlife collisions at RSW airport by 17%, by simply chasing them away.

Salt Lake City airport (SLC) has employed the use of pigs in order to disrupt the habitat of several Californian Gulls families in order to scare the birds away from the adjacent to the airport island, that severed as the bird's habitat.

All in all, so far there has been no proven, "best" way of preventing wildlife collisions, and perhaps there is no perfect way of totally stopping such collisions from happening, but, just as many personalities have said and many airports have done, there are ways of reducing such incidents from happening.

And from our understanding, some good ways of reducing the number of collisions are:

- Removing seed-bearing plants from the airport vicinity;
- Covering nearby water sources with netting;
- Removing brush and trees from airport vicinity;
- Using various devices that mimic predator behavior and loud noises to disturb wildlife;
- Employing the help of animals, such as dogs, pigs and predator birds, to disrupt any unwanted wildlife traffic;
- Employing the help of radars and staff members to spot and warn tower control, of any dangerous wildlife traffic;
- Thorough preparation and training of flight participants, both pilots and flight assistants, to better communicate and deal with any situation.

5. Conclusion

a. Denouement

In conclusion, during this semester we managed to not only gain knowledge about various Big Data techniques and a new programming language (python), but also a completely new part in the Programming world, a part which combines both

programming and business into a concept which helps companies all around the world to learn about their customers, about what they are doing well and what not, and helping them making informed decisions instead of guessing over gut feelings.

Our project turned out to be quite close to what we imagined when we were pitching the idea. We learned a lot of interesting facts regarding planes, birds and airports, while researching the matter. We also strengthened the knowledge gained during the courses and Digital Days event.

Although there is room for improvement, for example: The Predictive analysis could be further improved by using other datasets, like: bird flight patterns, weather patterns and a much bigger dataset of both successful and crashed flights; we are satisfied with what we achieved.

Due to Hand-in website limits, files like: datasets, powerBi projects and more, cannot be uploaded. To see how we worked and what files we created, one has to follow the link, which will take you to our GitHub repository: https://github.com/RaidenRabit/WASP

As an ending note, we would like to thank all the readers, who invested their time in reading this paper, also the guiding teacher, who helped and guided us throughout the entire process. All files used in the creation of this report are attached to the hand-in folder, in case you would like to inspect them in great detail.

b. References

- Diagram taken from “Balancing Agile with Discipline” by Barry Boehm Richard Turner
- Dataset for collisions: https://wildlife.faa.gov/databaseSearch.aspx
- Used to normalize data: http://aircharterguide.com/Operators
- Dataset for all flights: https://www.kaggle.com/freddejn/flights/data
- Info about airports and flight operators: https://openflights.org/data.html
- How to find Nr. of clusters: http://www.awesomestats.in/python-cluster-validation/
- How to use unsupervised learning: https://datascience.stackexchange.com/questions/16700/confused-about-how-to-apply-kmeans-on-my-a-dataset-with-features-extracted
- What state airport is in: https://data.humdata.org/dataset/ourairports
- Info about bird strikes: https://wildlife.faa.gov/downloads/StrikeReport1990-2012.pdf

- Busiest airports: https://www.faa.gov/airports/planning_capacity/passenger_allcargo_stats/passenger/media/cy14-commercial-service-enplanements.pdf
- Kaufman, Kenn (1996). Lives of North American Birds. Houghton Mifflin. p. 293. ISBN 0-395-77017-3.
- Mourning dove heat map: https://mnbirdatlas.org/species/mourning-dove/
- Collision distribution over aircraft: http://www.boeing.com/commercial/aeromagazine/articles/2011_q3/4/