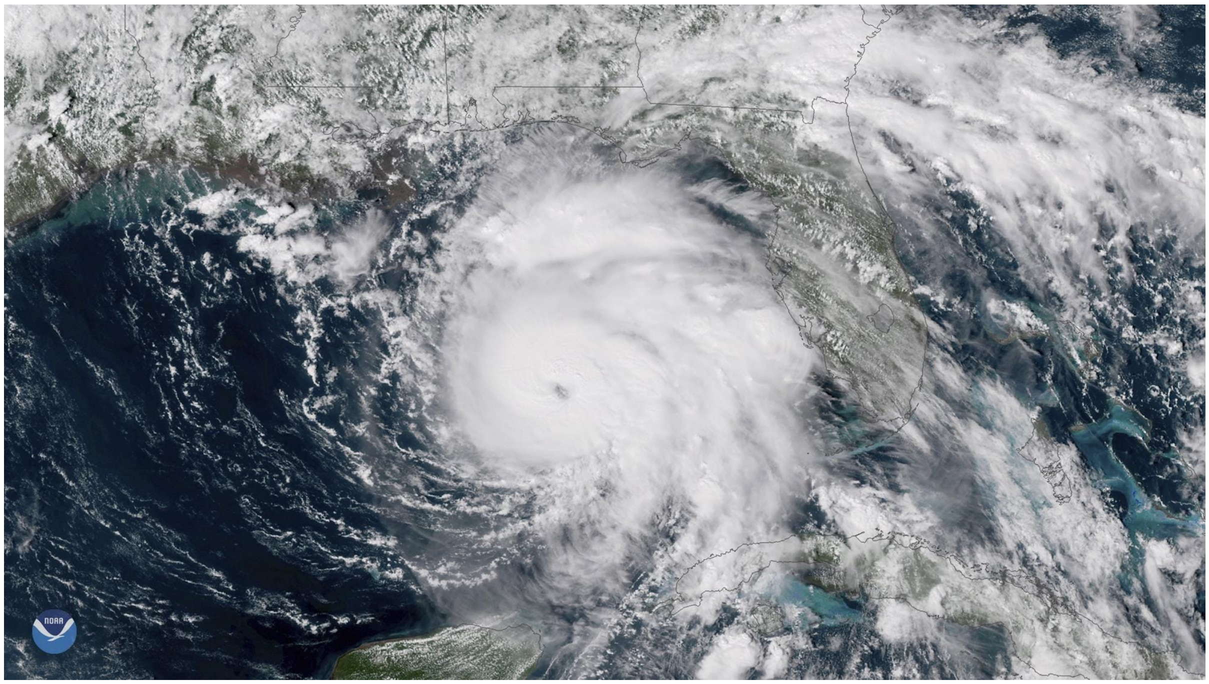
Hurricane Katrina Study

by Asad Imam and Michael Burrell



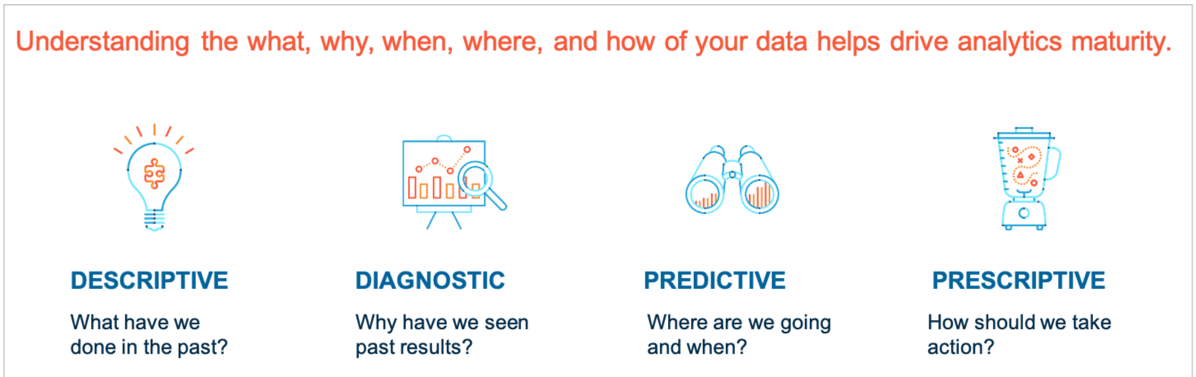
Abstract

In this study, we looked at several of the most devastating hurricanes to hit the United States, such as Hurricane Katrina. We set out to predict what factors (features) associated with hurricanes lead to the most damage (USD) using machine learning models including linear regression, decision trees/random forests, support vector machines, and artificial neural networks. Our results included selection of the Random Forest Tree Ensemble model with an normalized mean residual squared error (NMRSE) of 0.084.

Introduction

Although planet Earth is a beautiful home, we are constantly reminded of the annual occurrence of natural disasters, Hurricanes being one of the world's most recurrent. Each hurricane season (June - November), the United States is hit with about ten storms; 6 of which become hurricanes and about 2 of which become major hurricanes (Category 3 or greater). These hurricanes are responsible for the highest number of deaths (6,593) from 1980-2020 and they cause enormous amounts of damage. Of the 258 U.S. weather disasters since 1980, tropical cyclones have caused the most damage at $945 billion total, with an average cost of almost $21.5 billion per event.

The main objective of our project is to analyze historical Hurricane data from numerous sources in order to understand why some hurricanes, such as Katrina in 2005, are more devastating than others by identifying which variables are best for predicting hurricane damage using machine learning. A major component to why Hurricane Katrina was so devastating is because of the extremely high storm surges that were reported, which led to the breaking of the levees and floodwalls. Storm Surge is measured as feet above the normal tide and is a common statistic used for Hurricane reporting. It is our hypothesis that the higher the storm surge of a hurricane, the higher the damage costs will be. We will explore this problem using a methodology following the 4 Types of Data Analytics; Descriptive Analysis, Diagnostic Analysis, Predictive Analysis, Prescriptive Analysis.



The use of descriptive analytics will help identify certain trends, characteristics, and relationships in the data. We then will use diagnostic analytics to discover why we have seen past results using predictive models on our hypothesized predictor (storm surge) From this, we can then use the same predictive modeling techniques to predict the damage that current and future hurricanes will have based on the features selected. Furthermore, we aim to use prescriptive analytics to help to prepare for such severe impacts, as Katrina caused. This includes faster severe hurricane detection and building smarter infrastructure for the future.

About the Data Set

For our data set we initially considered 30 of the deadliest hurricanes in US History. This data was based on the list published by CBS News article [Deadliest hurricanes in U.S. history (https://www.cbsnews.com/pictures/deadliest-hurricanes-worst-in-the-us-list/)](https://www.cbsnews.com/pictures/deadliest-hurricanes-worst-in-the-us-list/). We then added 20 more hurricanes which were also Category 3 or above from the National Hurricane Center's (NHC) National Oceanic and Atmospheric Administration (NOAA) Website. [Link to Website (https://www.nhc.noaa.gov/data/tcr/index.php?season=2005&basin=atl)](https://www.nhc.noaa.gov/data/tcr/index.php?season=2005&basin=atl). We collected the following data about the hurricanes

1. **Name** : Name of the Hurricane
2. **Minimum Elevation** : Refers to the Minimum Elevation of the place worst impacted by the Hurricane. This is relative to the sea level (0ft).
3. **Category**: The Category of the Hurricane. (Ranges from 1-5: The hurricanes we considered were all atleast Category 3).
4. **Year** : Year when the Hurricane occured.
5. **Month** : Month of the Hurricane. (Majority of the Hurricane when hurricane lasted more than a month)
6. **Damage (2021 USD)**: The Damage caused by the Hurricane. This amount is in 2021 USD.
7. **Casualties**: The number of people that died from the Hurricane.
8. **Days Long**: The number of days the Hurricane Lasted.
9. **Minimum Pressure**: The minimum pressure of the hurricane. This is computed in mb.

10. **Peak Wind Speed**: The Peak Wind Speed of the Hurricane. This is computed in mph.
11. **Max Storm Surge**: The Storm Surge of the Hurricane. This is measured in ft.
    Note: Storm Surge is measured as the rise in water level above the normal tidal level.

Data Preprocessing

In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
df = pd.read_excel('NHC Historical Storm Data.xlsx')
```

In [3]:
```python
df.head()
```

Out[3]:

	ID	Name	Minimum Elevation (ft)	Category	Year	Month	Damage (2021 USD)	Casualties	Days Long
0	NaN	Great Galveston	7	4	1900	September	1.156021e+09	8000	20
1	NaN	Maria	4	5	2017	September	1.007176e+11	3057	17
2	NaN	Okeechobee	4	5	1928	September	1.604152e+09	2500	16
3	AL122005	Katrina	-8	5	2005	August	2.247291e+11	1833	9
4	NaN	Chenière Caminada	-8	4	1893	October	1.523944e+08	2000	9

Finding out how many rows and columns our dataset has

In [4]:
```python
df.shape
```

Out[4]:
```
(50, 12)
```

Dropping the ID Column since it doesn't provide any information about the hurricanes

In [5]:
```python
df = df.drop('ID', 1)
```

In [6]:
```python
df.columns
```

Out[6]:
```
Index(['Name', 'Minimum Elevation (ft)', 'Category', 'Year', 'Month',
       'Damage (2021  USD)', 'Casualties', 'Days Long',
       'Minimum Pressure (mb)', 'Peak Wind Speed (mph)',
       'Max Storm Surge (ft)'],
      dtype='object')
```

We don't have ID in the dataset now

Removing missing values from the data frame

In [7]:
```
df.isna().sum()
```

Out[7]:
```
Name                       0
Minimum Elevation (ft)     0
Category                   0
Year                       0
Month                      0
Damage (2021  USD)         1
Casualties                 0
Days Long                  0
Minimum Pressure (mb)      0
Peak Wind Speed (mph)      0
Max Storm Surge (ft)       0
dtype: int64
```

We have 1 missing value in the Damage (2021 USD) column. Removing that Record

In [8]:
```
df.dropna(axis=0,inplace = True)
```

In [9]:
```
##Looking at the shape of our dataset now
df.shape
```

Out[9]:
```
(49, 11)
```

We removed the 1 missing record from our dataset

Normalizing the Damage (2021 USD)

In [10]:
```
#Looking at the damage
df['Damage (2021  USD)'].min()
```

Out[10]:
```
5836382.98
```

The smallest Damage value is 5.8 million. We will normalize the damage column by dividing each record by 1,000,000

In [11]:
```
df['Damage (2021  USD)']=df['Damage (2021  USD)']/1000000
```

In [12]:
```
print("the minimum normalized damage is", df['Damage (2021  USD)'].min(
```

```
the minimum normalized damage is 5.836382980000001  and the maximum norma
lized damage is  224729.134665
```

Looking at the 5 smallest records for damage

In [13]:
```python
df.nsmallest(5, ['Damage (2021  USD)'])
```

Out[13]:

	Name	Minimum Elevation (ft)	Category	Year	Month	Damage (2021 USD)	Casualties	Days Long	Minimum Pressure (mb)	Peak Wind Speed (mph)
21	1886 Indianola	95	2	1886	August	5.836383	150	19	925	150
5	Sea Islands	0	3	1893	August	30.478889	2000	18	954	185
30	Atlantic-Gulf	0	4	1999	September	36.223409	750	12	927	149
18	Georgia	0	4	1898	September	49.574096	179	11	938	130
19	1875 Indianola	95	3	1875	September	99.749091	800	10	955	115

We see that 1886 Indianola is an outlier for damage with 5 million dollars in damage. We will remove this record

In [14]:
```python
df = df.drop(21)
```

Looking at the 5 smallest records for damage again

In [15]:
```python
df.nsmallest(5, ['Damage (2021  USD)'])
```

Out[15]:

	Name	Minimum Elevation (ft)	Category	Year	Month	Damage (2021 USD)	Casualties	Days Long	Minimum Pressure (mb)	Peal Winc Speec (mph
5	Sea Islands	0	3	1893	August	30.478889	2000	18	954	18
30	Atlantic-Gulf	0	4	1999	September	36.223409	750	12	927	14
18	Georgia	0	4	1898	September	49.574096	179	11	938	13
19	1875 Indianola	95	3	1875	September	99.749091	800	10	955	11
20	1906 Florida Keys	2	3	1906	October	126.030206	240	15	953	12

The outlier Damage has now been removed from the dataset

Normalizing the Dataset

Normalizing Storm Surge

In [16]:
```
df['Max Storm Surge (ft)'] = (df['Max Storm Surge (ft)']-df['Max Storm
```

Normalizing Wind Speed

In [17]:
```
df['Peak Wind Speed (mph)'] = (df['Peak Wind Speed (mph)']-df['Peak Win
```

Normalizing Damage

In [18]:
```
df['Damage (2021  USD)'] = (df['Damage (2021  USD)']-df['Damage (2021
```

Normalizing Minimum Elevation

In [19]:
```
df['Minimum Elevation (ft)'] = (df['Minimum Elevation (ft)']-df['Minimu
```

Normalizing Pressure

In [20]:
```
df['Minimum Pressure (mb)'] = (df['Minimum Pressure (mb)']-df['Minimum
```

Normalizing Casualties

In [21]:
```
df['Casualties'] = (df['Casualties']-df['Casualties'].mean())/df['Casua
```

Viewing the complete normalized dataset

In [22]:
```
df.head(48)
```
Out[22]:

	Name	Minimum Elevation (ft)	Category	Year	Month	Damage (2021 USD)	Casualties	Days Long	Minimum Pressure (mb)	
0	Great Galveston	0.272917	4	1900	September	-0.391140	2.269092	20	-0.142631	-0.2
1	Maria	0.134048	5	2017	September	2.279546	0.635993	17	-1.132973	1.2
2	Okeechobee	0.134048	5	1928	September	-0.379120	0.451968	16	-0.228747	0.6
3	Katrina	-0.421430	5	2005	August	5.606090	0.231601	9	-1.391323	1.2
4	Chenière Caminada	-0.421430	4	1893	October	-0.418062	0.286775	9	0.589361	-0.4
5	Sea Islands	-0.051112	3	1893	August	-0.421333	0.286775	18	0.847711	1.6
6	San Ciriaco	-0.051112	4	1899	August	-0.404419	0.899641	31	-0.185689	0.2
7	Audrey	-0.421430	3	1957	June	-0.383657	-0.236556	5	0.503244	-0.8
8	Great Labor Day	0.041468	5	1935	September	-0.368440	-0.239200	12	-1.821906	1.6
10	Great Miami	-0.051112	4	1926	September	-0.380578	-0.251093	12	-0.185689	0.2
11	Grand Isle	-0.051112	3	1909	September	-0.413256	-0.258362	10	0.761594	-1.0
12	1919 Florida Keys	0.041468	4	1919	September	-0.412793	-0.118939	13	-0.314864	0.2
13	New Orleans	-0.421430	4	1915	September	-0.412679	-0.283141	10	-0.142631	0.0
14	Galveston	0.272917	4	1915	August	-0.385723	-0.283141	19	0.244894	-0.6
15	Camille	-0.051112	5	1969	August	-0.172182	-0.289418	8	-1.477439	1.2
16	New England	-0.051112	5	1938	September	-0.262461	-0.289418	14	0.244894	0.6
17	Diane	-0.051112	2	1955	August	-0.193798	-0.313206	16	1.493586	-1.6
18	Georgia	-0.051112	4	1898	September	-0.420820	-0.314858	11	0.158778	-0.6
19	1875 Indianola	4.346419	3	1875	September	-0.419474	-0.109688	10	0.890769	-1.2
20	1906 Florida Keys	0.041468	3	1906	October	-0.418769	-0.294704	15	0.804653	-1.0
22	Mississippi	-0.051112	3	1906	September	-0.406435	-0.329725	11	0.804653	-1.0
23	Cedar Keys	-0.051112	3	1896	September	-0.413741	-0.307259	8	1.106061	-0.8
24	Agnes	-0.051112	1	1972	June	-0.052479	-0.331708	12	1.838053	-2.4
25	Hazel	2.217088	4	1954	October	-0.317658	-0.178739	13	0.158778	-0.4
26	Betsy	-0.051112	4	1965	September	-0.090446	-0.347236	16	0.331011	-0.3
27	Great Atlantic	-0.051112	4	1944	September	-0.380342	-0.352852	7	-0.056514	0.0
28	Carol	-0.051112	3	1954	August	-0.295775	-0.350209	6	0.890769	-1.2

	Name	Minimum Elevation (ft)	Category	Year	Month	Damage (2021 USD)	Casualties	Days Long	Minimum Pressure (mb)	
29	Floyd	-0.051112	4	1999	September	-0.135065	-0.345253	12	-0.573214	0.4
30	Atlantic-Gulf	-0.051112	4	1999	September	-0.421178	-0.126207	12	-0.314864	0.1
31	Donna	-0.051112	4	1960	August	-0.178533	-0.253737	15	-0.185689	-0.0
32	Alicia	-4.680091	3	1983	August	-0.200517	-0.367059	6	1.192178	-1.2
33	Gilbert	-0.051112	5	1988	September	-0.236795	-0.268934	11	-1.994139	1.5
34	Hugo 1989	-0.051112	5	1989	September	0.230595	-0.353844	14	-0.702389	0.7
35	Andrew 1992	-0.051112	5	1992	August	1.009635	-0.352522	12	-0.530156	1.2
36	Opal 1995	-0.051112	4	1995	October	-0.195223	-0.353183	10	-0.788506	-0.0
37	Mitch 1998	0.550656	5	1998	October	-0.147684	6.010715	17	-1.262148	1.3
38	Keith 2000	-0.051112	4	2000	September	-0.408519	-0.351531	9	0.201836	-0.3
39	Iris 2001	-0.051112	4	2001	October	-0.411763	-0.362103	5	0.589361	-0.0
40	Isabel 2003	-0.051112	5	2004	September	-0.281919	-0.357147	14	-0.831564	0.9
41	Charley 2004	-0.051112	4	2004	August	0.212784	-0.369041	7	1.622761	0.2
42	Frances 2004	-0.051112	4	2004	August	0.045286	-0.357808	18	0.029603	0.0
43	Ivan 2004	-0.051112	5	2004	September	0.376386	-0.333360	24	0.374069	0.8
44	Jeanne 2004	-0.051112	3	2005	September	-0.391221	0.628725	17	0.675478	-1.0
45	Dennis 2005	-0.051112	4	2005	July	-0.338132	-0.344923	15	-0.185689	-0.9
46	Rita 2005	-0.051112	5	2005	September	-0.026547	-0.334351	9	-1.692731	1.2
47	Wilma 2005	-0.051112	5	2005	October	0.421804	-0.345253	13	-2.252489	1.5
48	Hurricane Ike 2008	-0.051112	4	2008	September	-0.319622	-0.309572	15	2.225578	-1.4
49	Ida	-0.051112	4	2021	August	2.126175	-0.336003	10	-0.185689	0.2

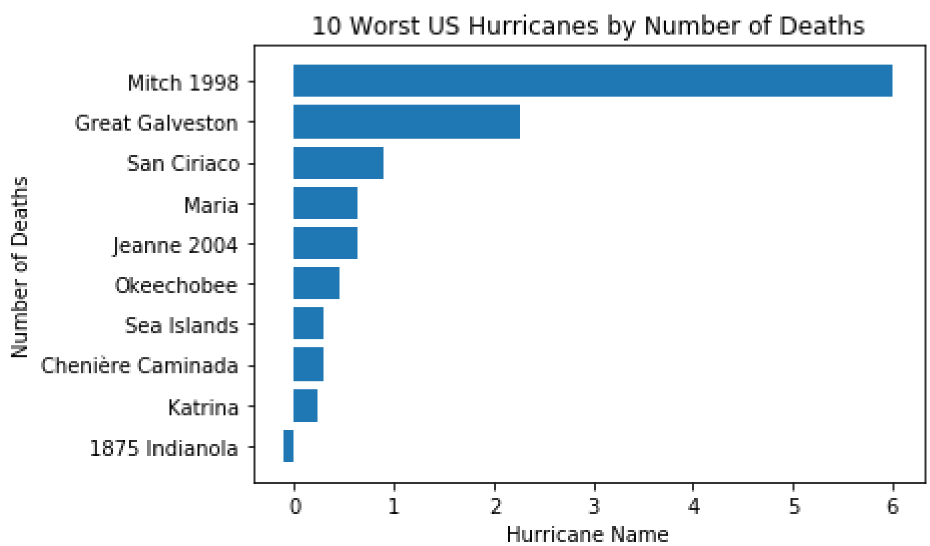
Descriptive Analytics

Descriptive analytics is the interpretation of historical data to better understand changes that have occurred in a business. Descriptive analytics describes the use of a range of historic data to draw comparisons.

Worst US Hurricanes by Number of Deaths

In [23]:
```
worst_casualties = df.sort_values('Casualties',ascending=False)[:10]
worst_casualties.sort_values('Casualties',ascending=True, inplace = Tru
```

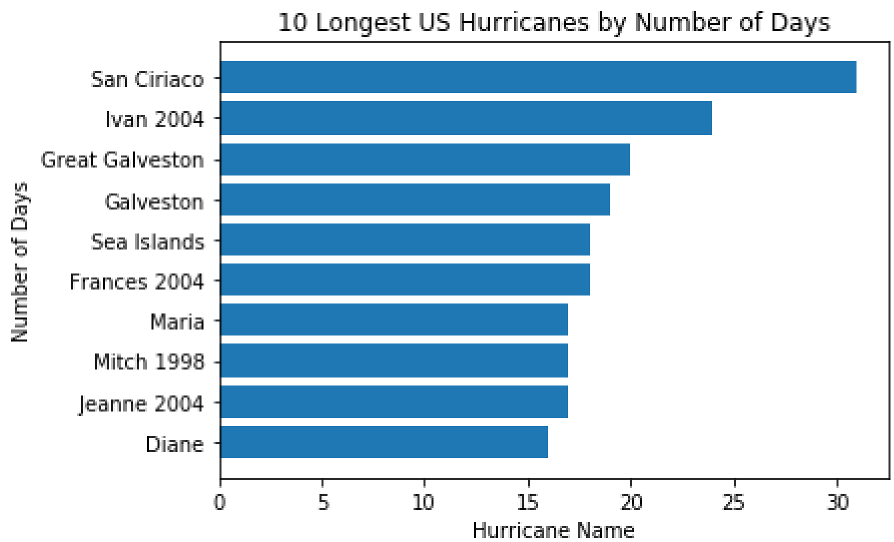
In [24]:
```
plt.barh(worst_casualties['Name'],worst_casualties['Casualties'])
plt.title('10 Worst US Hurricanes by Number of Deaths')
plt.xlabel('Hurricane Name')
plt.ylabel('Number of Deaths')
plt.show()
```



Longest US Hurricanes by Number of Days

In [25]:
```
longest_hurricanes = df.sort_values('Days Long',ascending=False)[:10]
longest_hurricanes.sort_values('Days Long',ascending=True, inplace = Tr
```

In [26]:
```
plt.barh(longest_hurricanes['Name'],longest_hurricanes['Days Long'])
plt.title('10 Longest US Hurricanes by Number of Days')
plt.xlabel('Hurricane Name')
plt.ylabel('Number of Days')
plt.show()
```

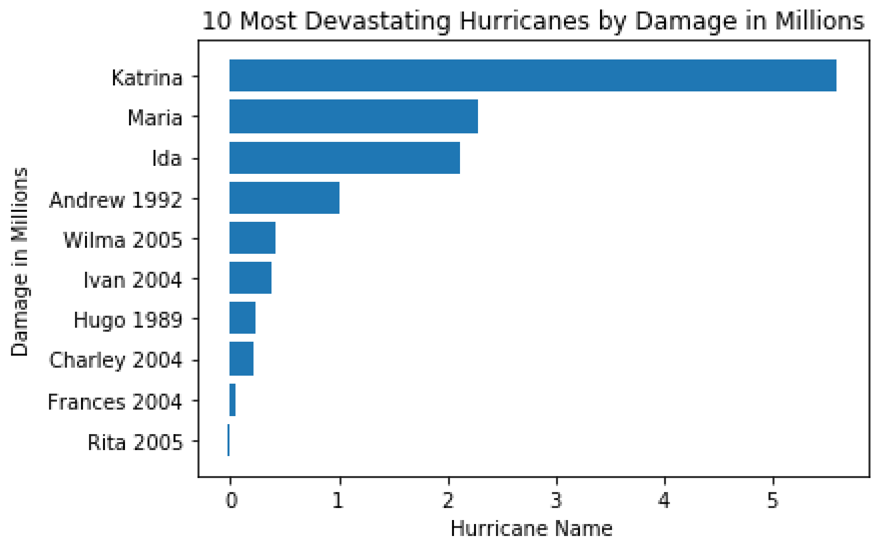


US Hurricanes by Damage in US Dollars 2021

In [27]:
```
expense_hurricanes = df.sort_values('Damage (2021  USD)',ascending=Fals
expense_hurricanes.sort_values('Damage (2021  USD)',ascending=True, inp
```

In [28]:

```
plt.barh(expense_hurricanes['Name'],expense_hurricanes['Damage (2021  U
plt.title('10 Most Devastating Hurricanes by Damage in Millions')
plt.xlabel('Hurricane Name')
plt.ylabel('Damage in Millions')
plt.show()
```



Here we can see that Hurricane Katrina was the most devastating hurricane in terms of damage

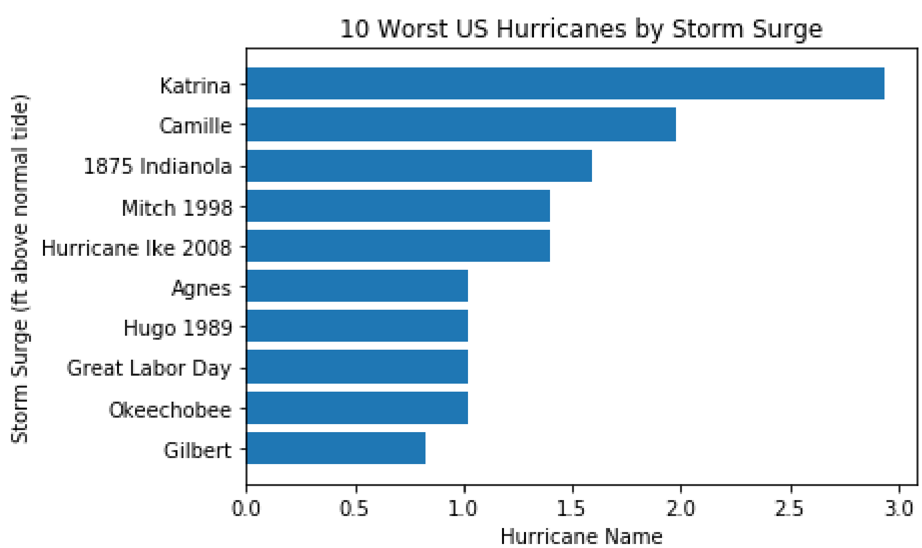
Worst US Hurricanes by Storm Surge

In [29]:

```
surge_hurricanes = df.sort_values('Max Storm Surge (ft)',ascending=Fals
surge_hurricanes.sort_values('Max Storm Surge (ft)',ascending=True, inp
```

In [30]:

```
plt.barh(surge_hurricanes['Name'],surge_hurricanes['Max Storm Surge (ft
plt.title('10 Worst US Hurricanes by Storm Surge')
plt.xlabel('Hurricane Name')
plt.ylabel('Storm Surge (ft above normal tide)')
plt.show()
```

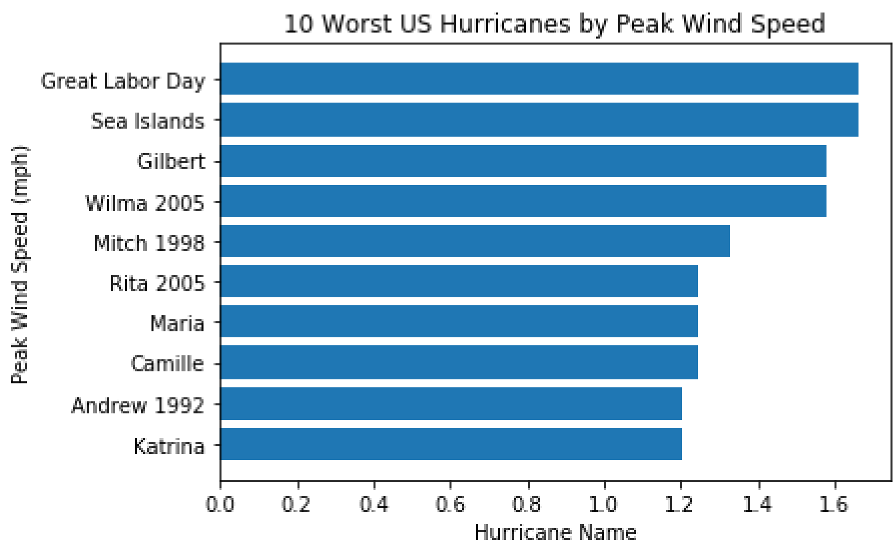


Here we can see that Hurricane Katrina had the highest storm surge

Worst US Hurricanes by Windspeed

In [31]:
```
wind_hurricanes = df.sort_values('Peak Wind Speed (mph)',ascending=Fals
wind_hurricanes.sort_values('Peak Wind Speed (mph)',ascending=True, inp
```

In [32]:
```
plt.barh(wind_hurricanes['Name'],wind_hurricanes['Peak Wind Speed (mph)
plt.title('10 Worst US Hurricanes by Peak Wind Speed')
plt.xlabel('Hurricane Name')
plt.ylabel('Peak Wind Speed (mph)')
plt.show()
```



Descriptive Statistics

In [33]:
```
df.describe()
```

Out[33]:

	Minimum Elevation (ft)	Category	Year	Damage (2021 USD)	Casualties	Days Long	Minimun Pressure (mb
count	4.800000e+01	48.000000	48.000000	4.800000e+01	4.800000e+01	48.000000	4.800000e+0
mean	3.642919e-17	3.979167	1960.666667	9.251859e-18	-1.734723e-17	12.770833	-9.830100e-1
std	1.000000e+00	0.887012	43.471846	1.000000e+00	1.000000e+00	4.938945	1.000000e+0
min	-4.680091e+00	1.000000	1875.000000	-4.213325e-01	-3.690413e-01	5.000000	-2.252489e+0
25%	-5.111165e-02	3.750000	1918.000000	-4.069563e-01	-3.457491e-01	9.750000	-5.409204e-0
50%	-5.111165e-02	4.000000	1967.000000	-3.186398e-01	-3.009818e-01	12.000000	-1.345573e-0
75%	-5.111165e-02	5.000000	2001.750000	-1.239101e-01	-1.656060e-01	15.250000	6.970069e-0
max	4.346419e+00	5.000000	2021.000000	5.606090e+00	6.010715e+00	31.000000	2.225578e+0

Moments

Moments [of a statistical distribution] The shape of any distribution can be described by its various 'moments'. The first four are:

1) The mean, which indicates the central tendency of a distribution.

2) The second moment is the variance, which indicates the width or deviation.

3) The third moment is the skewness, which indicates any asymmetric 'leaning' to either left or right.

4) The fourth moment is the Kurtosis, which indicates the degree of central 'peakedness' or, equivalently, the 'fatness' of the outer tails.

In [34]:

```
df.mean()
```

Out[34]:

```
Minimum Elevation (ft)    3.642919e-17
Category                  3.979167e+00
Year                      1.960667e+03
Damage (2021  USD)        9.251859e-18
Casualties               -1.734723e-17
Days Long                 1.277083e+01
Minimum Pressure (mb)    -9.830100e-18
Peak Wind Speed (mph)    -3.978299e-16
Max Storm Surge (ft)     -1.671117e-16
dtype: float64
```

In [35]:

```
df.var()
```

Out[35]:

```
Minimum Elevation (ft)       1.000000
Category                     0.786791
Year                      1889.801418
Damage (2021  USD)           1.000000
Casualties                   1.000000
Days Long                   24.393174
Minimum Pressure (mb)        1.000000
Peak Wind Speed (mph)        1.000000
Max Storm Surge (ft)         1.000000
dtype: float64
```

In [36]:

```
df.skew()
```

Out[36]:

```
Minimum Elevation (ft)   -0.212971
Category                 -0.912646
Year                     -0.371926
Damage (2021  USD)        4.377213
Casualties                5.084860
Days Long                 1.152828
Minimum Pressure (mb)    -0.202676
Peak Wind Speed (mph)    -0.110173
Max Storm Surge (ft)      0.389234
dtype: float64
```

In [37]:

```
df.kurtosis()
```

Out[37]:

```
Minimum Elevation (ft)    17.614028
Category                   1.426798
Year                      -1.349028
Damage (2021  USD)        21.866154
Casualties                29.029357
Days Long                  2.865794
Minimum Pressure (mb)     -0.066244
Peak Wind Speed (mph)     -0.566672
Max Storm Surge (ft)       0.572863
dtype: float64
```

Diagnostic Analytics

Diagnostic analytics is a form of advanced analytics that examines data or content to answer the question, “Why did it happen?” It is characterized by techniques such as drill-down, data discovery, data mining and correlations.

Correlation Matrix

In [38]:

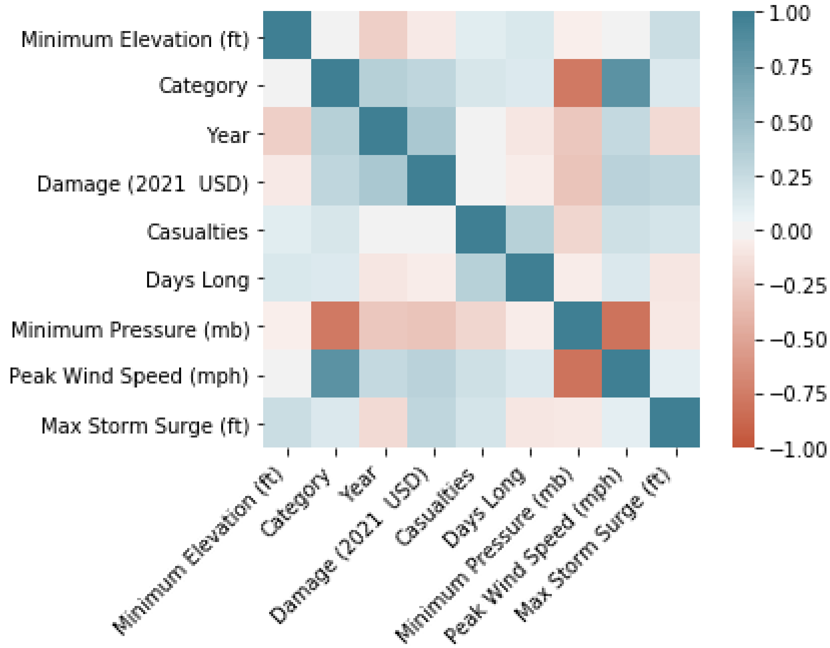
```
corr = df.corr()
corr
```

Out[38]:

	Minimum Elevation (ft)	Category	Year	Damage (2021 USD)	Casualties	Days Long	Minimum Pressure (mb)	Peak Wind Speed (mph)
Minimum Elevation (ft)	1.000000	0.030095	-0.238460	-0.071556	0.110923	0.149988	-0.046375	0.004795
Category	0.030095	1.000000	0.346333	0.296432	0.170940	0.139730	-0.759842	0.832920
Year	-0.238460	0.346333	1.000000	0.400277	-0.015717	-0.100947	-0.283216	0.262652
Damage (2021 USD)	-0.071556	0.296432	0.400277	1.000000	0.024506	-0.054897	-0.302486	0.322284
Casualties	0.110923	0.170940	-0.015717	0.024506	1.000000	0.330484	-0.195756	0.214146
Days Long	0.149988	0.139730	-0.100947	-0.054897	0.330484	1.000000	-0.052042	0.143251
Minimum Pressure (mb)	-0.046375	-0.759842	-0.283216	-0.302486	-0.195756	-0.052042	1.000000	-0.803915
Peak Wind Speed (mph)	0.004795	0.832920	0.262652	0.322284	0.214146	0.143251	-0.803915	1.000000
Max Storm Surge (ft)	0.237121	0.141595	-0.157653	0.291497	0.185826	-0.094577	-0.082176	0.091811

In [39]:

```
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```



Result: For Diagnostic Analytics, we will predict the value of Storm Surge based on Features such as Minimum Pressure, Minimum Elevation and Peak Wind Speed

Multiple Linear Regression

Using Minimum Pressure and Wind Speed to predict the Storm Surge

In [40]:
```
import statsmodels.api as sm
```

In [41]:
```
X = df[["Minimum Pressure (mb)","Peak Wind Speed (mph)"]]
y = df["Max Storm Surge (ft)"]

# Note the difference in argument order
model = sm.OLS(y, X).fit()
predictions = model.predict(X) # make the predictions by the model

# Print out the statistics
model.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Max Storm Surge (ft)	**R-squared (uncentered):**	0.009
Model:	OLS	**Adj. R-squared (uncentered):**	-0.034
Method:	Least Squares	**F-statistic:**	0.2002
Date:	Mon, 20 Dec 2021	**Prob (F-statistic):**	0.819
Time:	21:55:55	**Log-Likelihood:**	-67.396
No. Observations:	48	**AIC:**	138.8
Df Residuals:	46	**BIC:**	142.5
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Minimum Pressure (mb)	-0.0237	0.247	-0.096	0.924	-0.521	0.473
Peak Wind Speed (mph)	0.0728	0.247	0.295	0.769	-0.424	0.570

Omnibus:	1.482	**Durbin-Watson:**	1.357
Prob(Omnibus):	0.477	**Jarque-Bera (JB):**	0.771
Skew:	0.270	**Prob(JB):**	0.680
Kurtosis:	3.306	**Cond. No.**	3.03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Using Minimum Elevation (ft),Minimum Pressure (mb) and Peak Wind Speed (mph) to predict Surge

In [42]:

```
X = df[["Minimum Elevation (ft)","Minimum Pressure (mb)","Peak Wind Spe
y = df["Max Storm Surge (ft)"]

# Note the difference in argument order
model = sm.OLS(y, X).fit()
predictions = model.predict(X) # make the predictions by the model

# Print out the statistics
model.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	Max Storm Surge (ft)	**R-squared (uncentered):**	0.064
Model:	OLS	**Adj. R-squared (uncentered):**	0.002
Method:	Least Squares	**F-statistic:**	1.033
Date:	Mon, 20 Dec 2021	**Prob (F-statistic):**	0.387
Time:	21:55:55	**Log-Likelihood:**	-66.005
No. Observations:	48	**AIC:**	138.0
Df Residuals:	45	**BIC:**	143.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Minimum Elevation (ft)	0.2369	0.145	1.639	0.108	-0.054	0.528
Minimum Pressure (mb)	0.0048	0.243	0.020	0.984	-0.485	0.494
Peak Wind Speed (mph)	0.0946	0.243	0.389	0.699	-0.394	0.584

Omnibus:	2.551	**Durbin-Watson:**	1.363
Prob(Omnibus):	0.279	**Jarque-Bera (JB):**	1.598
Skew:	0.270	**Prob(JB):**	0.450
Kurtosis:	3.712	**Cond. No.**	3.04

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Computing Errors

Root Mean Squared Error **(RMSE)** and Mean Absolute Error **(MAE)** are metrics used to evaluate a Regression Model. These metrics tell us how accurate our predictions are and, what is the amount of deviation from the actual values.

Technically, RMSE is the Root of the Mean of the Square of Errors and MAE is the Mean of Absolute value of Errors. Here, errors are the differences between the predicted values (values predicted by our regression model) and the actual values of a variable. They are calculated as follows :

$$\text{RMSE} = \sqrt{\frac{\sum (y_i - y_p)^2}{n}}$$

$$\text{MAE} = \frac{|(y_i - y_p)|}{n}$$

y_i = actual value
y_p = predicted value
n = number of observations/rows

In [43]:
```python
#RMSE/MAE computation using sklearn library
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [44]:
```python
multiple_regression_rmse = round(np.sqrt(mean_squared_error(y, predicti
multiple_regression_mae = round(mean_absolute_error(y, predictions),3)
print(f'The root mean square error (RMSE) for our multiple regression m
print(f'The mean absolute error (MAE) for our multiple regression model
```

```
The root mean square error (RMSE) for our multiple regression model is:
0.957
The mean absolute error (MAE) for our multiple regression model is: 0.719
```

The Normalized Root Mean Square Error **(NRMSE)** the RMSE facilitates the comparison between models with different scales. the normalised RMSE (NRMSE) which relates the RMSE to the observed range of the variable. Thus, the NRMSE can be interpreted as a fraction of the overall range that is typically resolved by the model.

There are multiple ways of Normalizing RMSE. See below:

You can normalize by

- the **mean**: $NRMSE = \frac{RMSE}{\bar{y}}$ (similar to the CV and applied in *INDperform*)
- the **difference between maximum and minimum**: $NRMSE = \frac{RMSE}{y_{max} - y_{min}}$,
- the **standard deviation**: $NRMSE = \frac{RMSE}{\sigma}$, or
- the **interquartile range**; $NRMSE = \frac{RMSE}{Q1 - Q3}$, i.e. the difference between 25th and 75th percentile,

of observations.

We will use the difference between the maximum and minimum to normalize the RMSE

In [45]:
```python
multiple_regression_nrmse = round(multiple_regression_rmse/(y.max()-y.m
print(f'The normalized root mean square error (NRMSE) for our multiple
```

```
The normalized root mean square error (NRMSE) for our multiple regression
model is: 0.199
```

Support Vector Machine

In [46]:
```
#Importing the libraries
from sklearn.svm import SVR
```

Predicting Surge using Minimum Elevation (ft),Minimum Pressure (mb) and Peak Wind Speed (mph) using SVM

In [47]:
```
X = df[["Minimum Elevation (ft)","Minimum Pressure (mb)","Peak Wind Spe
y = df[["Max Storm Surge (ft)"]]
```

In [48]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

In [49]:
```
#Fitting the Support Vector Regression Model to the dataset
regressor = SVR()
regressor.fit(X,y)
```

```
/Users/asadimam270/opt/anaconda3/lib/python3.7/site-packages/sklearn/util
s/validation.py:760: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[49]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scal
e',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [50]:
```
predictions = regressor.predict(X)
```

In [51]:
```
svm_rmse = round(np.sqrt(mean_squared_error(y, predictions)),3)
svm_mae = round(mean_absolute_error(y, predictions),3)
svm_nrmse = round(svm_rmse/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our model is: {svm_rmse}'
print(f'The mean absolute error (MAE) for our model is: {svm_mae}')
print(f'The normalized root mean square error (NRMSE) for our model is:
```

```
The root mean square error (RMSE) for our model is: 0.895
The mean absolute error (MAE) for our model is: 0.624
The normalized root mean square error (NRMSE) for our model is: 0.184
```

Tuning the Parameters

In [52]:
```
#Fitting the Support Vector Regression Model to the dataset
regressor = SVR(C=1000, cache_size=200, coef0=0.0, degree=3, epsilon=0.
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False
regressor.fit(X,y)
#Predict
predictions = regressor.predict(X)
#Computing errors
svm_rmse = round(np.sqrt(mean_squared_error(y, predictions)),3)
svm_mae = round(mean_absolute_error(y, predictions),3)
svm_nrmse = round(svm_rmse/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our model is: {svm_rmse}'
print(f'The mean absolute error (MAE) for our model is: {svm_mae}')
print(f'The normalized root mean square error (NRMSE) for our model is:
```

```
The root mean square error (RMSE) for our model is: 0.636
The mean absolute error (MAE) for our model is: 0.323
The normalized root mean square error (NRMSE) for our model is: 0.131

/Users/asadimam270/opt/anaconda3/lib/python3.7/site-packages/sklearn/util
s/validation.py:760: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Decision Trees

Predicting Surge using Minimum Elevation (ft),Minimum Pressure (mb) and Peak Wind Speed (mph) using Decision Tree Model

In [53]:
```
# Import the necessary modules and libraries
from sklearn.tree import DecisionTreeRegressor
```

In [54]:
```
X = df[["Minimum Elevation (ft)","Minimum Pressure (mb)","Peak Wind Spe
y = df[["Max Storm Surge (ft)"]]
```

In [55]:
```
# Fitting the Decision Tree Regression model
dt_regr = DecisionTreeRegressor()
dt_regr.fit(X, y)
```

Out[55]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', 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='deprecated',
                      random_state=None, splitter='best')
```

In [56]:
```
# Predict
y_predict = dt_regr.predict(X)
```

In [57]:
```
dt_rmse = round(np.sqrt(mean_squared_error(y, y_predict)),3)
dt_mae = round(mean_absolute_error(y, y_predict),3)
dt_nrmse = round(dt_rmse/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our model is: {dt_rmse}')
print(f'The mean absolute error (MAE) for our model is: {dt_mae}')
print(f'The normalized root mean square error (NRMSE) for our model is:
```

```
The root mean square error (RMSE) for our model is: 0.126
The mean absolute error (MAE) for our model is: 0.029
The normalized root mean square error (NRMSE) for our model is: Max Storm
Surge (ft)    0.026
dtype: float64
```

Artificial Neural Networks

Predicting Surge using Minimum Elevation (ft),Minimum Pressure (mb) and Peak Wind Speed (mph) using ANN

In [58]:
```
X = df[["Minimum Elevation (ft)","Minimum Pressure (mb)","Peak Wind Spe
y = df[["Max Storm Surge (ft)"]]
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
```

In [59]:

```
#Importing the libraries
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
model = Sequential()

# Adding the input layer and the first hidden layer
model.add(Dense(6, activation = 'relu', input_dim = 3))

# Adding the output layer
model.add(Dense(units = 1))

model.compile(loss = 'mean_squared_error',metrics = ["accuracy"])

# Training the model
model1 = model.fit(X, y, batch_size = 10, epochs = 100)

# Testing the Model
y_pred = model.predict(X)
```

```
Epoch 1/100
5/5 [==============================] - 1s 1ms/step - loss: 1.1948 - accur
acy: 0.0000e+00
Epoch 2/100
5/5 [==============================] - 0s 1ms/step - loss: 1.1701 - accur
acy: 0.0000e+00
Epoch 3/100
5/5 [==============================] - 0s 1ms/step - loss: 1.1526 - accur
acy: 0.0000e+00
Epoch 4/100
5/5 [==============================] - 0s 1ms/step - loss: 1.1372 - accur
acy: 0.0000e+00
Epoch 5/100
5/5 [==============================] - 0s 2ms/step - loss: 1.1246 - accur
acy: 0.0000e+00
Epoch 6/100
5/5 [==============================] - 0s 1ms/step - loss: 1.1135 - accur
acy: 0.0000e+00
Epoch 7/100
```

In [60]:

```
ann_rmse = round(np.sqrt(mean_squared_error(y, y_pred)),3)
ann_mae = round(mean_absolute_error(y, y_pred),3)
ann_nrmse = round(ann_rmse/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our ANN model is: {ann_rm
print(f'The mean absolute error (MAE) for our ANN model is: {ann_mae}')
print(f'The normalized root mean square error (NRMSE) for our ANN model
```

```
The root mean square error (RMSE) for our ANN model is: 0.904
The mean absolute error (MAE) for our ANN model is: 0.697
The normalized root mean square error (NRMSE) for our ANN model is: Max S
torm Surge (ft)    0.188
dtype: float64
```

Model Selection

In [61]:

```
print("Errors for SVM")
print(f'RMSE : {svm_rmse}')
print(f'MAE : {svm_mae}')
print(f'NRMSE : {svm_nrmse}')
print('')
print("Errors for Decision Tree")
print(f'RMSE : {dt_rmse}')
print(f'MAE : {dt_mae}')
print(f'NRMSE : {dt_nrmse}')
print('')
print("Errors for ANN")
print(f'RMSE : {ann_rmse}')
print(f'MAE : {ann_mae}')
print(f'NRMSE : {ann_nrmse}')
```

```
Errors for SVM
RMSE : 0.636
MAE : 0.323
NRMSE : 0.131

Errors for Decision Tree
RMSE : 0.126
MAE : 0.029
NRMSE : Max Storm Surge (ft)    0.026
dtype: float64

Errors for ANN
RMSE : 0.904
MAE : 0.697
NRMSE : Max Storm Surge (ft)    0.188
dtype: float64
```

In [62]:

```python
# import module
from tabulate import tabulate

# assign data
mydata = [["Decision Tree", "RMSE",dt_rmse,0.89],
          ["Decision Tree", "MAE",dt_mae,0.67],
          ["Decision Tree", "RMSE",dt_nrmse,1.85e-17],
          ["SVM", "RMSE",svm_rmse,0.96],
          ["SVM", "MAE",svm_mae,0.70],
          ["SVM", "RMSE",svm_nrmse,0.11],
          ["ANN", "RMSE",ann_rmse,0.96],
          ["ANN", "MAE",ann_mae,0.72],
          ["ANN", "RMSE",ann_nrmse,0.1]]

# create header
head = ["Model", "Measure","Error in Python", "Error in R"]

# display table
print('Errors for our models in Python and R:')
print(tabulate(mydata, headers=head, tablefmt="grid"))
```

```
Errors for our models in Python and R:
+---------------+-----------+-------------------+--------------+
| Model         | Measure   |   Error in Python |   Error in R |
+===============+===========+===================+==============+
| Decision Tree | RMSE      |             0.126 |     0.89     |
+---------------+-----------+-------------------+--------------+
| Decision Tree | MAE       |             0.029 |     0.67     |
+---------------+-----------+-------------------+--------------+
| Decision Tree | RMSE      |             0.026 |     1.85e-17 |
+---------------+-----------+-------------------+--------------+
| SVM           | RMSE      |             0.636 |     0.96     |
+---------------+-----------+-------------------+--------------+
| SVM           | MAE       |             0.323 |     0.7      |
+---------------+-----------+-------------------+--------------+
| SVM           | RMSE      |             0.131 |     0.11     |
+---------------+-----------+-------------------+--------------+
| ANN           | RMSE      |             0.904 |     0.96     |
+---------------+-----------+-------------------+--------------+
| ANN           | MAE       |             0.697 |     0.72     |
+---------------+-----------+-------------------+--------------+
| ANN           | RMSE      |             0.188 |     0.1      |
+---------------+-----------+-------------------+--------------+
```

Result: From the errors above we can see that Decision Tree was our best model to predict the Storm Surge

Predictive Analytics

Predictive analytics encompasses a variety of statistical techniques from data mining, predictive modelling, and machine learning that analyze current and historical facts to make predictions about future or otherwise unknown events.

We will be using the same models as above to predict the Damage based on Storm Surge, Minimum Elevation, Minimum Pressure and Peak Wind Speed

Support Vector Machines

In [63]:
```
X = df[["Max Storm Surge (ft)","Minimum Elevation (ft)","Minimum Pressu
y = df["Damage (2021  USD)"]
```

In [64]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y.values.reshape(-1,1))
```

In [65]:
```
#Fitting the Support Vector Regression Model to the dataset
regressor = SVR()
regressor.fit(X,y)
```

```
/Users/asadimam270/opt/anaconda3/lib/python3.7/site-packages/sklearn/util
s/validation.py:760: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[65]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scal
e',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [66]:
```
predictions = regressor.predict(X)
```

In [67]:

```python
svm_rmse_pred = round(np.sqrt(mean_squared_error(y, predictions)),3)
svm_mae_pred = round(mean_absolute_error(y, predictions),3)
svm_nrmse_pred = round(svm_rmse_pred/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our model is: {svm_rmse_p
print(f'The mean absolute error (MAE) for our model is: {svm_mae_pred}'
print(f'The normalized root mean square error (NRMSE) for our model is:
```

```
The root mean square error (RMSE) for our model is: 0.934
The mean absolute error (MAE) for our model is: 0.341
The normalized root mean square error (NRMSE) for our model is: 0.153
```

Artificial Neural Networks

In [68]:

```python
X = df[["Max Storm Surge (ft)","Minimum Elevation (ft)","Minimum Pressu
y = df["Damage (2021  USD)"]
```

In [69]:

```python
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
```

In [70]:

```
#Importing the libraries
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
model = Sequential()

# Adding the input layer and the first hidden layer
model.add(Dense(8, activation = 'relu', input_dim = 4))

# Adding the output layer
model.add(Dense(units = 1))

model.compile(loss = 'mean_squared_error',metrics=["accuracy"])

# Training the model
model1 = model.fit(X, y, batch_size = 10, epochs = 100)

# Testing the Model
y_pred = model.predict(X)
```

```
Epoch 1/100
5/5 [==============================] - 0s 1ms/step - loss: 1.0580 - accur
acy: 0.0000e+00
Epoch 2/100
5/5 [==============================] - 0s 1ms/step - loss: 1.0257 - accur
acy: 0.0000e+00
Epoch 3/100
5/5 [==============================] - 0s 1ms/step - loss: 1.0062 - accur
acy: 0.0000e+00
Epoch 4/100
5/5 [==============================] - 0s 1ms/step - loss: 0.9895 - accur
acy: 0.0000e+00
Epoch 5/100
5/5 [==============================] - 0s 1ms/step - loss: 0.9770 - accur
acy: 0.0000e+00
Epoch 6/100
5/5 [==============================] - 0s 1ms/step - loss: 0.9644 - accur
acy: 0.0000e+00
Epoch 7/100
```

In [71]:

```
ann_rmse_pred = round(np.sqrt(mean_squared_error(y, y_pred)),3)
ann_mae_pred = round(mean_absolute_error(y, y_pred),3)
ann_nrmse_pred = round(ann_rmse_pred/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our ANN model is: {ann_rm
print(f'The mean absolute error (MAE) for our ANN model is: {ann_mae_pr
print(f'The normalized root mean square error (RMSE) for our ANN model
```

```
The root mean square error (RMSE) for our ANN model is: 0.867
The mean absolute error (MAE) for our ANN model is: 0.459
The normalized root mean square error (RMSE) for our ANN model is: 0.459
```

Decision Tree

In [72]:
```
X = df[["Max Storm Surge (ft)","Minimum Elevation (ft)","Minimum Pressu
y = df["Damage (2021  USD)"]
```

In [73]:
```
dt_regr = DecisionTreeRegressor()
dt_regr.fit(X, y)
```

Out[73]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', 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='deprecated',
                      random_state=None, splitter='best')
```

In [74]:
```
# Predict
y_predict = dt_regr.predict(X)
```

In [75]:
```
dt_rmse_pred = np.sqrt(mean_squared_error(y, y_predict))
dt_mae_pred = mean_absolute_error(y, y_predict)
dt_nrmse_pred = dt_rmse_pred/(y.max()-y.min())
print(f'The root mean square error (RMSE) for our model is: {dt_rmse_pr
print(f'The mean absolute error (MAE) for our model is: {dt_mae_pred}')
print(f'The normalized root mean square error (NRMSE) for our model is:
```

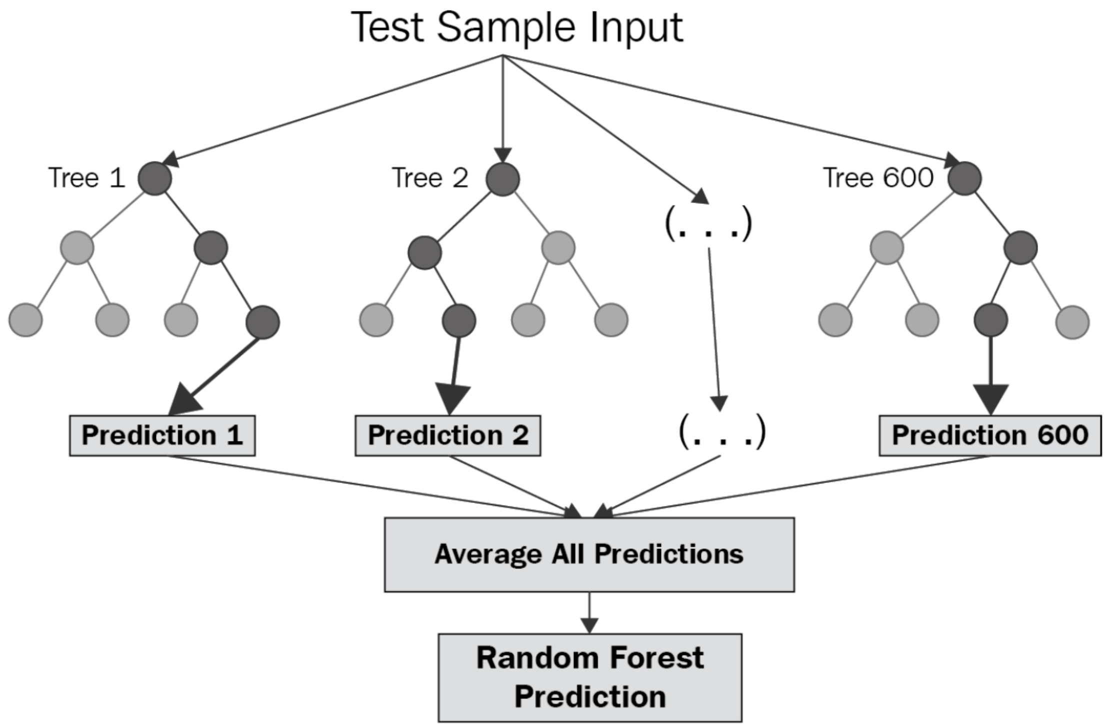
```
The root mean square error (RMSE) for our model is: 4.951616820006514e-05
The mean absolute error (MAE) for our model is: 1.0107445508998844e-05
The normalized root mean square error (NRMSE) for our model is: 8.2151480
71977484e-06
```

Result: It seems that our Decision Tree Model is overfitting. We will use the Tree Ensemble Method instead.

Theory: "Decision trees carry a big risk of overfitting, and tend to find local optima because they can't go back after they have made a split. To address these weaknesses, we turn to Random Forest, which illustrates the power of combining many decision trees into one model."
This info was taken from the Article below: [Random Forest Regression (https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f)](https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f)

Ensemble Method: Ensemble means a group of elements viewed as a whole rather than individually. An Ensemble method creates multiple models and combines them to solve it. Ensemble methods help to improve the robustness/generalizability of the model.

Random Forrest



Random Forest Structure

In [76]:
```
X = df[["Max Storm Surge (ft)","Minimum Elevation (ft)","Minimum Pressu
y = df["Damage (2021  USD)"]
```

In [77]:
```
from sklearn.ensemble import RandomForestRegressor
```

For this model I've chosen 5 trees (n_estimator=5)

In [78]:
```
rf_pred = RandomForestRegressor(n_estimators=5)
```

In [79]:
```
#Training the Model
rf_pred.fit(X, y)
# Predict
y_predict = rf_pred.predict(X)
```

In [80]:
```
rf_rmse_pred = round(np.sqrt(mean_squared_error(y, y_predict)),3)
rf_mae_pred = round(mean_absolute_error(y, y_predict),3)
rf_nrmse_pred = round(rf_rmse_pred/(y.max()-y.min()),3)
print(f'The root mean square error (RMSE) for our model is: {rf_rmse_pr
print(f'The mean absolute error (MAE) for our model is: {rf_mae_pred}')
print(f'The normalized root mean square error (NRMSE) for our model is:
```

```
The root mean square error (RMSE) for our model is: 0.496
The mean absolute error (MAE) for our model is: 0.241
The normalized root mean square error (NRMSE) for our model is: 0.082
```

Model Selection

In [81]:

```python
# assign data
errors_pred = [
            ["SVM", "RMSE",svm_rmse_pred],
            ["SVM", "MAE",svm_mae_pred],
            ["SVM", "NRMSE",svm_nrmse_pred],
            ["ANN", "RMSE",ann_rmse_pred],
            ["ANN", "MAE",ann_mae_pred],
            ["ANN", "NRMSE",ann_nrmse_pred],
            ["Decision Tree", "RMSE",dt_rmse_pred],
            ["Decision Tree", "MAE",dt_mae_pred],
            ["Decision Tree", "NRMSE",dt_nrmse_pred],
            ["Random Forrest", "RMSE",rf_rmse_pred],
            ["Random Forrest", "MAE",rf_mae_pred],
            ["Random Forrest", "NRMSE",rf_nrmse_pred],

]

# create header
head = ["Model", "Measure","Error in Python"]

# display table
print(tabulate(errors_pred, headers=head, tablefmt="grid"))
```

```
+-----------------+-----------+--------------------+
| Model           | Measure   |    Error in Python |
+=================+===========+====================+
| SVM             | RMSE      |        0.934       |
+-----------------+-----------+--------------------+
| SVM             | MAE       |        0.341       |
+-----------------+-----------+--------------------+
| SVM             | NRMSE     |        0.153       |
+-----------------+-----------+--------------------+
| ANN             | RMSE      |        0.867       |
+-----------------+-----------+--------------------+
| ANN             | MAE       |        0.459       |
+-----------------+-----------+--------------------+
| ANN             | NRMSE     |        0.144       |
+-----------------+-----------+--------------------+
| Decision Tree   | RMSE      |        4.95162e-05 |
+-----------------+-----------+--------------------+
| Decision Tree   | MAE       |        1.01074e-05 |
+-----------------+-----------+--------------------+
| Decision Tree   | NRMSE     |        8.21515e-06 |
+-----------------+-----------+--------------------+
| Random Forrest  | RMSE      |        0.496       |
+-----------------+-----------+--------------------+
| Random Forrest  | MAE       |        0.241       |
+-----------------+-----------+--------------------+
| Random Forrest  | NRMSE     |        0.082       |
+-----------------+-----------+--------------------+
```

Results: Although Decision Trees returned the lowest errors, we suspect that this might be an issue of overfitting (which is a common error with Decision Tree). To resolve this we used an ensemble approach (namely Random Forrest). Based on the errors above we will say that Random Forrest was our best model for predicting the Damage.

Conclusion

The main goal of this study was to use machine learning algorithms to identify which variables or features are best for predicting hurricane damage. Through descriptive and diagnostic analytics, we found that Max Storm Surge, Minimum Elevation, Minimum Pressure, and Peak Wind Speed were the best predictors for our target, Damage. Our predictive analytics included modeling using linear regression, decision trees/random forests, support vector machines, and artificial neural networks. The selection of the Random Forest Tree Ensemble model with an normalized mean residual squared error (NMRSE) of 0.084 was our best predictive model, thus the best model to predict damage for any hurricane in history or the future.

Discussion

Limitations to our study included data availability and accuracy issues. Some of our hurricane records went back as far as the 1800s and there just isn't much data on hurricanes that far back. Additionally, we felt that the integrity of our data would be lost if we decided to add more features with inconsistent and/or inaccurate data. We see a great opportunity for future work on this study, especially on the addition of new, accurate, and insightful features. This can include but is not limited to percipitation, sea surface temperature, and tornadoes.

References

1. https://worldpopulationreview.com/state-rankings/states-with-the-least-natural-disasters (https://worldpopulationreview.com/state-rankings/states-with-the-least-natural-disasters)
2. https://coast.noaa.gov/states/fast-facts/hurricane-costs.html#:~:text=Of%20the%20258%20U.S.%20weather,6%2C593%20between%201980%2 (https://coast.noaa.gov/states/fast-facts/hurricane-costs.html#:~:text=Of%20the%20258%20U.S.%20weather,6%2C593%20between%201980%2
3. https://www.cnn.com/2021/04/13/weather/2021-atlantic-hurricane-season-fast-facts/index.html (https://www.cnn.com/2021/04/13/weather/2021-atlantic-hurricane-season-fast-facts/index.html)
4. https://www.cbsnews.com/pictures/deadliest-hurricanes-worst-in-the-us-list/ (https://www.cbsnews.com/pictures/deadliest-hurricanes-worst-in-the-us-list/)