CLIMATE CHANGE EARTH SURFACE TEMPERATURE

We are going to see the climate change around the world and the temperatuer difference on the earth through out the years. According to wikipedia Global warming, also referred to as climate change, is the observed century-scale rise in the average temperature of the Earth's climate system and its related effects.

The word is climate change is one of the biggest existential threat that humanity is facing. Hoping to throw some exploratory light on the matter with the given data.

EXPLORATORY DATA ANALYSIS

EDA

Importing all the necessary libraries

In [290]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/climate-change-earth-surface-temperature-data/GlobalTemperatures.csv
/kaggle/input/climate-change-earth-surface-temperature-data/GlobalLandTemperaturesByState.csv
/kaggle/input/climate-change-earth-surface-temperature-data/GlobalLandTemperaturesByCountry.csv
/kaggle/input/climate-change-earth-surface-temperature-data/GlobalLandTemperaturesByCity.csv
/kaggle/input/climate-change-earth-surface-temperature-data/GlobalLandTemperaturesByMajorCity.csv
```

In [291]:

```
Temp_1=pd.read_csv('../input/climate-change-earth-surface-te
mperature-data/GlobalTemperatures.csv')
```

In [292]:

```
Temp_1.columns
```

Out[292]:

```
Index(['dt', 'LandAverageTemperature', 'LandAverageTemperatu
reUncertainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty'
,
       'LandMinTemperature', 'LandMinTemperatureUncertainty'
,
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
      dtype='object')
```

In [293]:

```
Temp=Temp_1.copy()
```

In [294]:

```
Temp.shape
```

Out[294]:

```
(3192, 9)
```

OBSERVATION:

LandMaxTemperatureIt has 3192 rows and 9 columns

In [295]:

```
Temp.columns
```

Out[295]:

```
Index(['dt', 'LandAverageTemperature', 'LandAverageTemperatu
reUncertainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty'
,
       'LandMinTemperature', 'LandMinTemperatureUncertainty'
,
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
      dtype='object')
```

OBSERVATION:

The 9 columns are 1)dt 2)LandAverageTemparature 3) LandAverageTemparateUncertainty 4) LandMaxTemperature 5) LandMaxTemperatureUncertainty 6) LandMinTemperature 7) LandMinTemperatureUncertainty 8) LandAndOceanAverageTemperature 9) LandAndOceanAverageTemperatureUncertainty

In [296]:

```
Temp.head
```

Out[296]:

```
<bound method NDFrame.head of                dt  LandAverageT
emperature  LandAverageTemperatureUncertainty  \
0     1750-01-01                   3.034
3.574
1     1750-02-01                   3.083
3.702
2     1750-03-01                   5.626
3.076
3     1750-04-01                   8.490
2.451
4     1750-05-01                  11.573
2.072
...          ...                    ...
...
3187  2015-08-01                  14.755
0.072
3188  2015-09-01                  12.999
0.079
3189  2015-10-01                  10.801
0.102
3190  2015-11-01                   7.433
0.119
3191  2015-12-01                   5.518
0.100

      LandMaxTemperature  LandMaxTemperatureUncertainty  Lan
dMinTemperature  \
0                    NaN                            NaN
NaN
1                    NaN                            NaN
NaN
2                    NaN                            NaN
NaN
3                    NaN                            NaN
```

```
NaN
4                     NaN                              NaN
NaN
...                   ...                              ...
...
3187               20.699                            0.110
9.005
3188               18.845                            0.088
7.199
3189               16.450                            0.059
5.232
3190               12.892                            0.093
2.157
3191               10.725                            0.154
0.287

      LandMinTemperatureUncertainty  LandAndOceanAverageTemp
erature  \
0                               NaN
NaN
1                               NaN
NaN
2                               NaN
NaN
3                               NaN
NaN
4                               NaN
NaN
...                             ...
...
3187                          0.170
17.589
3188                          0.229
17.049
3189                          0.115
16.290
3190                          0.106
15.252
3191                          0.099
14.774

      LandAndOceanAverageTemperatureUncertainty
0                                           NaN
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN
...                                         ...
3187                                      0.057
3188                                      0.058
3189                                      0.062
```

```
3190                                        0.063
3191                                        0.062

[3192 rows x 9 columns]>
```

OBSERVATION:

It shows the first five rows of the file and all the 9 columns, in which some columns has null values.source of the website was also provided in the last column

Here we can observe that some columns has numerical values whereas some have strings

In [297]:

```
missing=Temp.isnull().sum()
```

In [298]:

```
missing
```

Out[298]:

```
dt                                              0
LandAverageTemperature                         12
LandAverageTemperatureUncertainty              12
LandMaxTemperature                           1200
LandMaxTemperatureUncertainty                1200
LandMinTemperature                           1200
LandMinTemperatureUncertainty                1200
LandAndOceanAverageTemperature               1200
LandAndOceanAverageTemperatureUncertainty    1200
dtype: int64
```

OBSERVATION:

we can observe that maximum all the columns has null values only one column i.e dt has no null values

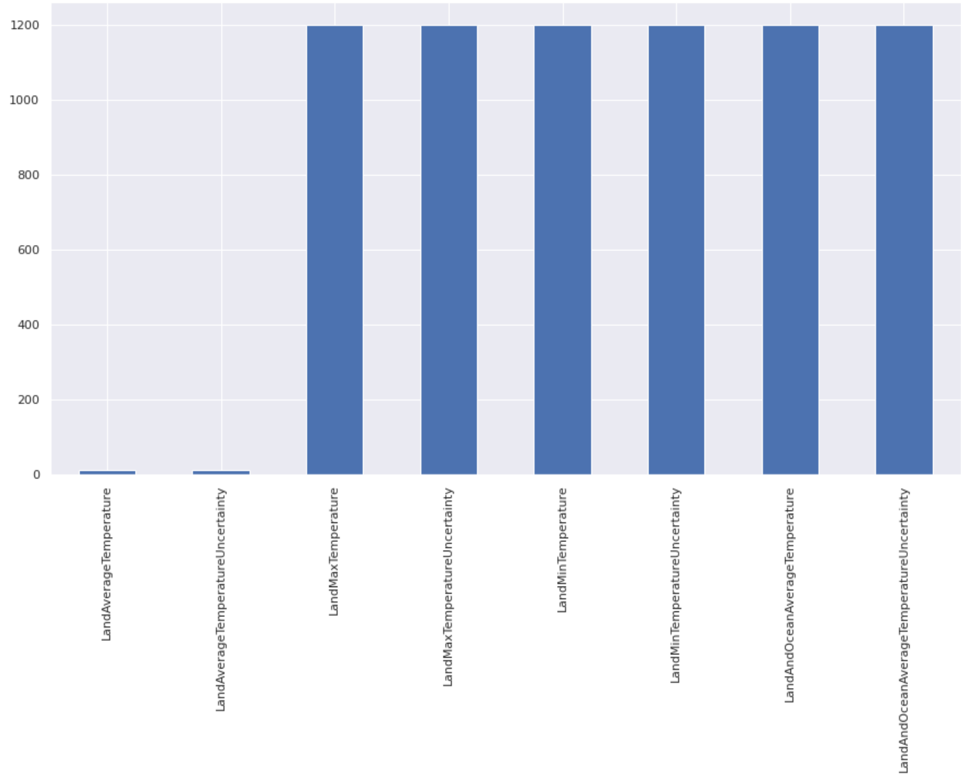
Remaining all the columns have null values. They need to be filled up with appropriate values later on

In [299]:

```
missing=missing[missing>0]
missing.sort_values(inplace=True)
plt.figure(figsize=(15,8))
missing.plot.bar()
```

Out[299]:

<AxesSubplot:>



OBSERVATION:

In the above graph we can observe that LandAverageTemparature and LandAverageTemparatureUncertainty has very less null values i.e. in range of 10-20 whereas remaining all the columns has equal null values I.e in tha range of 1100-1200

In [300]:

```
Temp.dtypes.value_counts()
```

Out[300]:

```
float64    8
```

```
object     1
dtype: int64
```

OBSERVATION:

8 columns are Floating point numbers and 1 columns are object data type which is Text or mixed numeric and non-numeric values

In [301]:

```
Temp['dt'].describe()
```

Out[301]:

```
count          3192
unique         3192
top      1750-01-01
freq              1
Name: dt, dtype: object
```

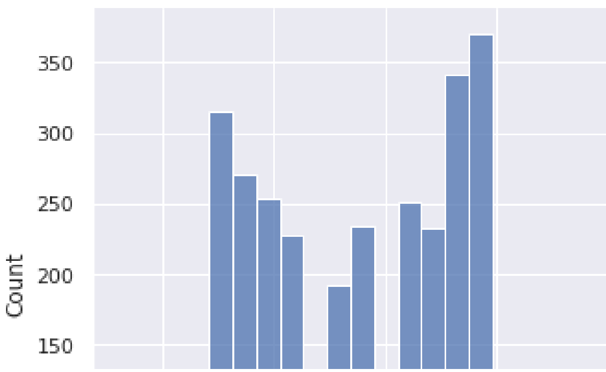
OBSERVATION:

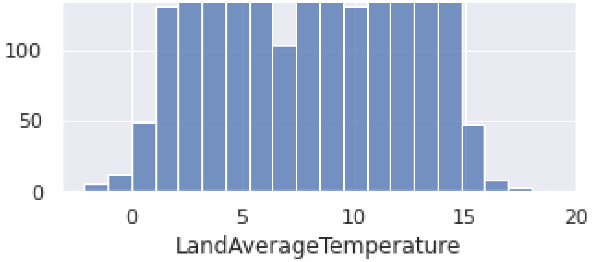
This returns Different stats like count of values, unique values, top and frequency of occurences in this case, top and name. Here the count of values are 3192 ,and it has 3192 unique values, and frequency of occurences are 1

HISTOGRAM

In [302]:

```
sns.set(rc={'figure.figsize':(12,8)})
sns.displot(Temp['LandAverageTemperature'], kde=False, bins=
20) ;
```





OBSERVATION:

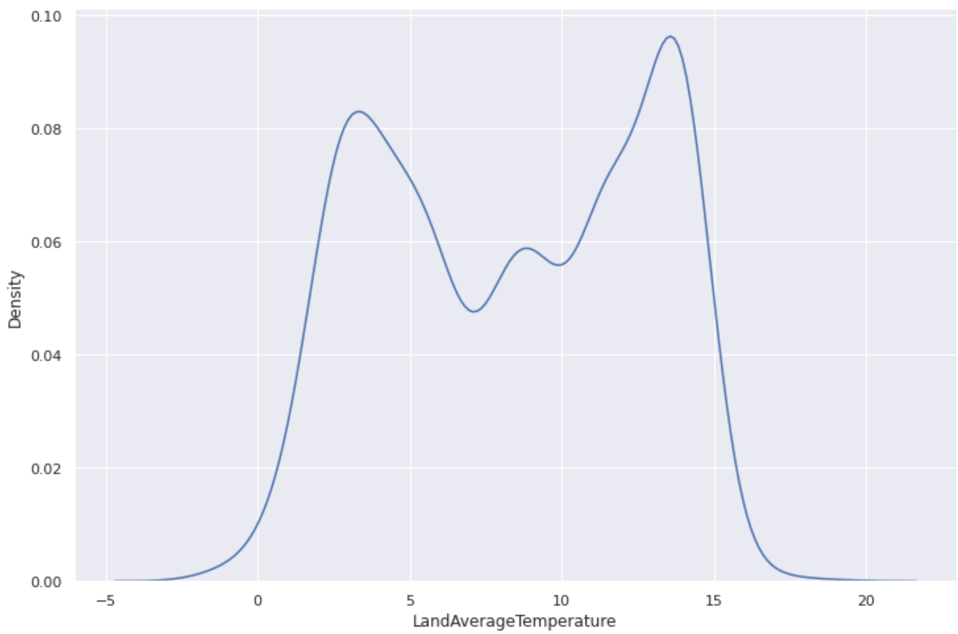
Here we can observe land average temperature, most of the countries average temperature is nearly 15 and we can observe in some countries land average temperature is less than 0

In [303]:

```
sns.kdeplot(Temp['LandAverageTemperature'])
```

Out[303]:

```
<AxesSubplot:xlabel='LandAverageTemperature', ylabel='Densit
y'>
```



OBSERVATION:

Here the peak point is between 13-15 and the tail part shows that very few states have vaccinations per hundreds

In [304]:

```
Temp['LandAverageTemperature'].describe()
```

Out[304]:

```
count    3180.000000
mean        8.374731
std         4.381310
min        -2.080000
25%         4.312000
50%         8.610500
75%        12.548250
max        19.021000
Name: LandAverageTemperature, dtype: float64
```

OBSERVATION:

Different stats were returned like count of values, mean, mode, minimum value, maximum value and standard deviation etc

SCATTERPLOT

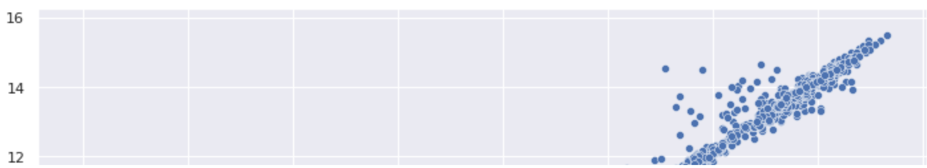
Scatter plots use a collection of points placed using Cartesian coordinates to display values from two variables. By displaying a variable in each axis, we can detect if a relationship or correlation between the two variables exists. Scatter Plots are also great for observing the spread of the data as they retain the exact data values and sample size.

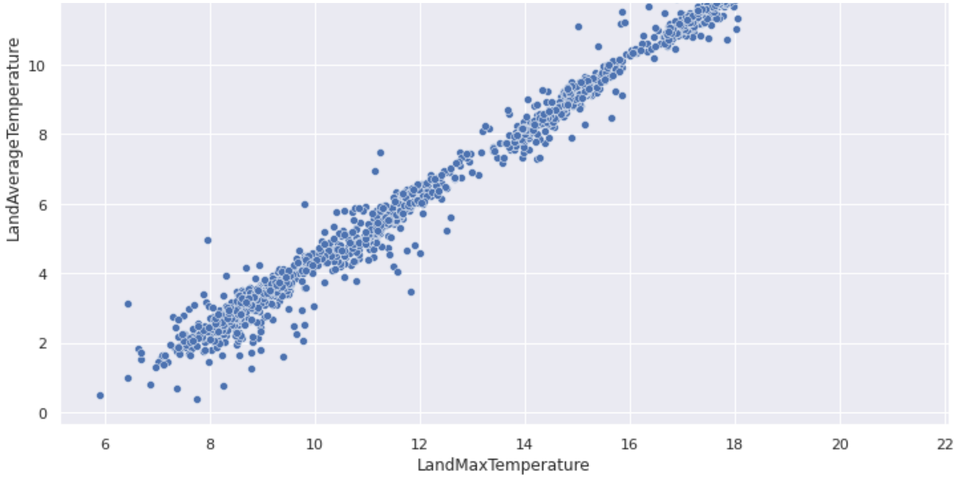
In [305]:

```
sns.scatterplot(x='LandMaxTemperature', y='LandAverageTemperature', data=Temp)
```

Out[305]:

```
<AxesSubplot:xlabel='LandMaxTemperature', ylabel='LandAverageTemperature'>
```





OBSERVATION:

This scatter plot shows LandMaxTemperature on x-axis and LandAverageTemparature on Y-axis.

CORRELATION

correlation can be calculated only on numerical columns we can't caluculate correlation on non-numeric

In [306]:

```
numeric_features = Temp.select_dtypes(include = [np.number]
)
numeric_features.columns
```

Out[306]:

```
Index(['LandAverageTemperature', 'LandAverageTemperatureUnce
rtainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty'
,
       'LandMinTemperature', 'LandMinTemperatureUncertainty'
,
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
      dtype='object')
```

OBSERVATION:

Here numeric columns are stored in variable called number if features and columns are extracted. We can see that nearly 8 columns out of 9 have numerical values

In [307]:

```
numeric_features1 = Temp.select_dtypes(exclude = ['O'])
numeric_features1.columns
```

Out[307]:

```
Index(['LandAverageTemperature', 'LandAverageTemperatureUnce
rtainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty'
,
       'LandMinTemperature', 'LandMinTemperatureUncertainty'
,
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
      dtype='object')
```

OBSERVATION:

Here numeric columns are extracted excluding strings

In [308]:

```
numeric_features.shape, numeric_features1.shape
```

Out[308]:

```
((3192, 8), (3192, 8))
```

OBSERVATION:

it gives number of numeric columns are there. We can get concluded that nearly 8 colums have numeric values

In [309]:

```
categorical_features = Temp.select_dtypes(include = [np.obj
ect])
categorical_features.columns
```

Out[309]:

```
Index(['dt'], dtype='object')
```

OBSERVATION:

It gives all the string columns. We have nearly 1 string columns

The one string column is date column

In [310]:

```
categorical_features.shape
```

Out[310]:

```
(3192, 1)
```

In [311]:

```
correlation = numeric_features.corr()
print(correlation['LandAverageTemperature'].sort_values(asce
nding = False), '\n')
```

```
LandAverageTemperature                        1.000000
LandMaxTemperature                            0.995807
LandMinTemperature                            0.995611
LandAndOceanAverageTemperature                0.988066
LandMaxTemperatureUncertainty                -0.108462
LandAndOceanAverageTemperatureUncertainty    -0.131412
LandMinTemperatureUncertainty                -0.167451
LandAverageTemperatureUncertainty            -0.204191
Name: LandAverageTemperature, dtype: float64
```

OBSERVATION:

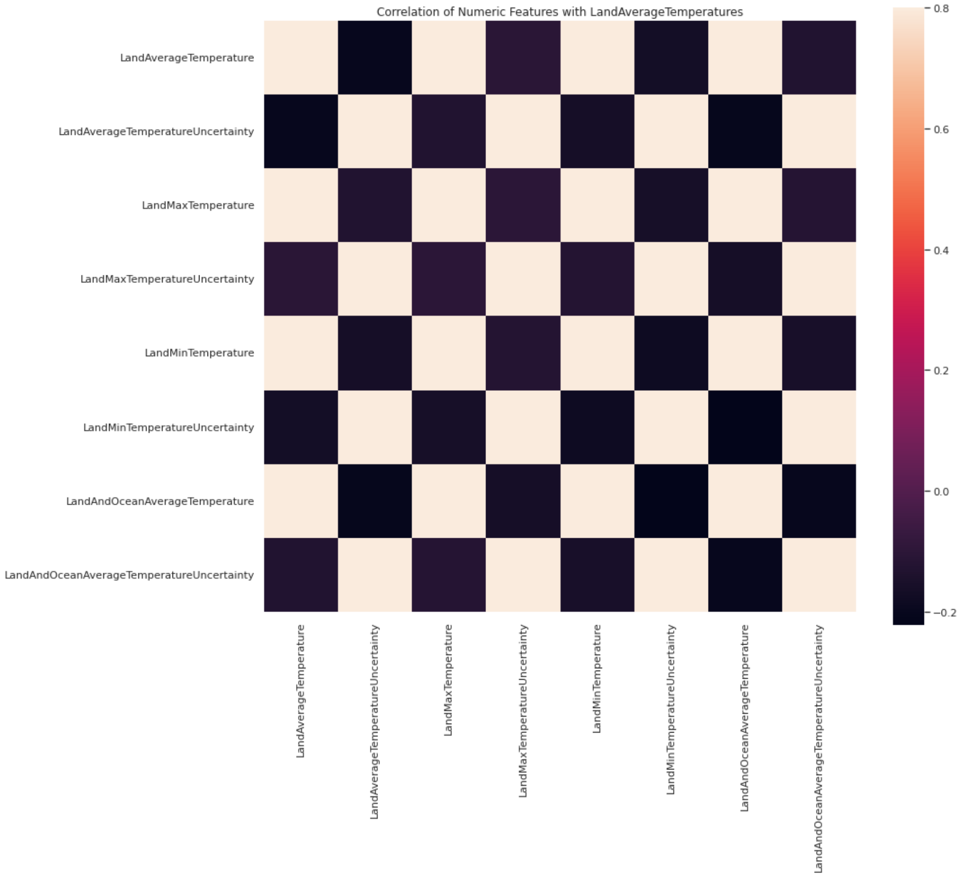
To find the correlation between numerical features we are using corr method

In [312]:

```
f, ax = plt.subplots(figsize = (14, 12))
plt.title('Correlation of Numeric Features with LandAverageT
emperatures')
sns.heatmap(correlation, square=True, vmax=0.8)
```

Out[312]:

```
<AxesSubplot:title={'center':'Correlation of Numeric Feature
s with LandAverageTemperatures'}>
```



OBSERVATION:

This is the correlation matrix for all the 9 numerical columns

In [313]:

```
k=5
cols = correlation.nlargest(k, 'LandAverageTemperature')['LandAverageTemperature'].index
print(cols)
```

```
Index(['LandAverageTemperature', 'LandMaxTemperature', 'LandMinTemperature',
       'LandAndOceanAverageTemperature', 'LandMaxTemperatureUncertainty'],
      dtype='object')
```

OBSERVATION:

it shows the 5 numerical columns

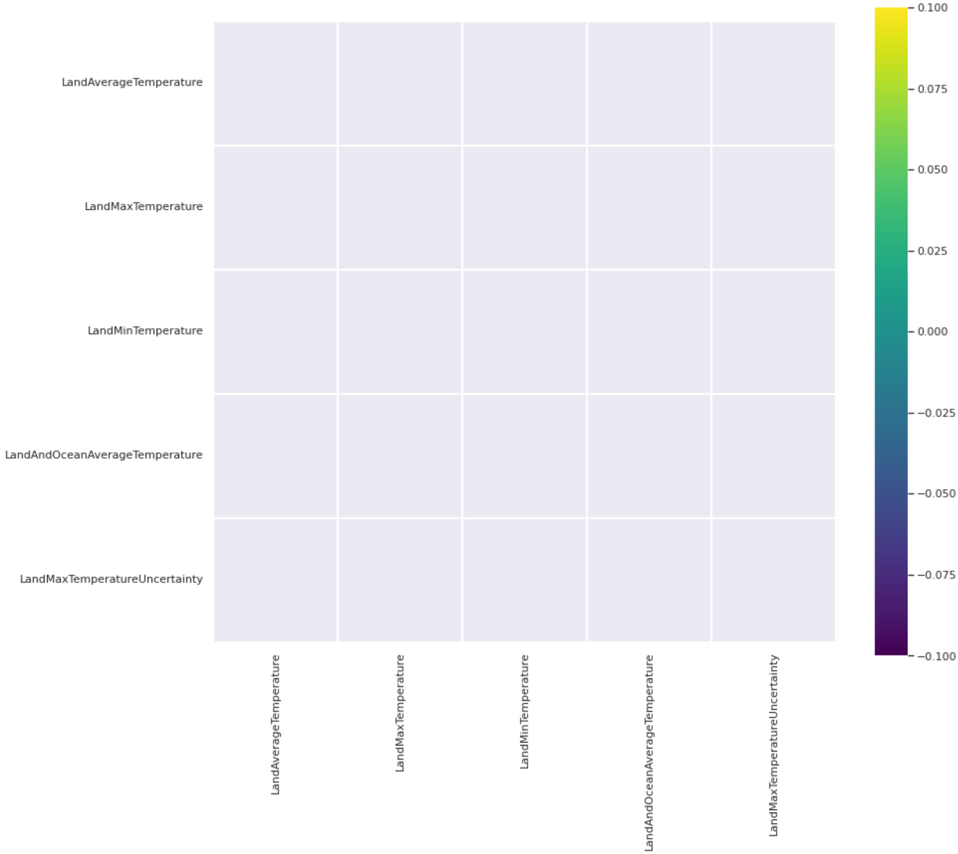
In [314]:

```
cm = np.corrcoef(Temp[cols].values.T)
f, ax = plt.subplots(figsize = (14, 12))
sns.heatmap(cm, vmax=0.8, linewidths=0.01, square=True, annot=True, cmap='viridis', linecolor='white', xticklabels=cols.values, yticklabels=cols.values)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/matrix.py:198
: RuntimeWarning: All-NaN slice encountered
  vmin = np.nanmin(calc_data)
```

Out[314]:

<AxesSubplot:>



OBSERVATION:

Here correlation matrix for top 5 columns was plotted

Here correlation matrix for top 5 columns was plotted

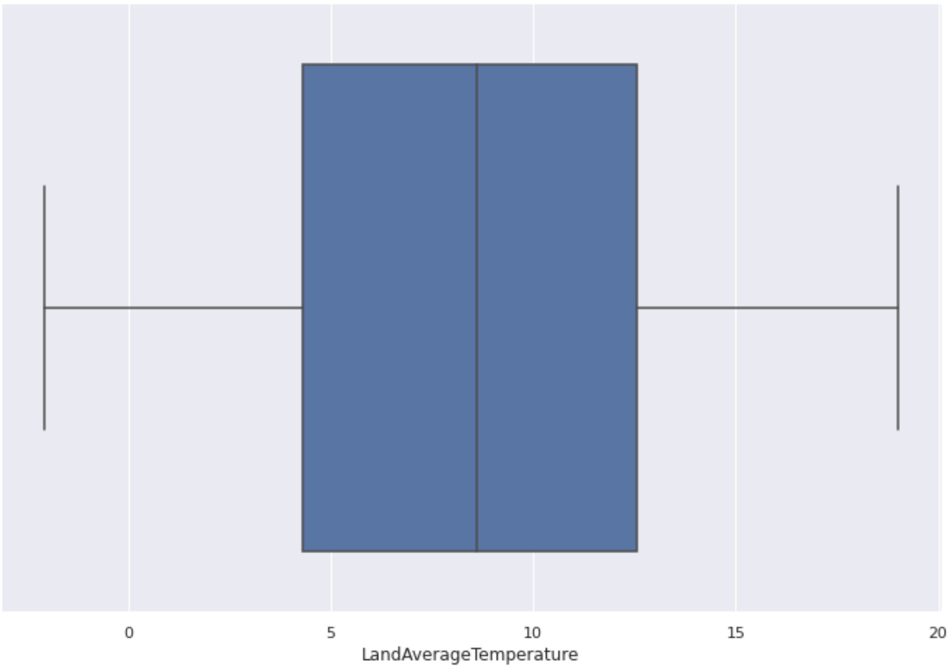
In [315]:

```
sns.boxplot(Temp['LandAverageTemperature'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.p
y:43: FutureWarning: Pass the following variable as a keywor
d arg: x. From version 0.12, the only valid positional argum
ent will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation
.
  FutureWarning
```

Out[315]:

```
<AxesSubplot:xlabel='LandAverageTemperature'>
```



It is the boxplot land average temperature we can observe that median of total vaccinations per hundred is between 10-20.Here we can observe some outliers which are not fitting the box we can remove those outliers to reduce the difference from mean to median

In [316]:

```
f, ax = plt.subplots(figsize = (16,10))
fig = sns.boxplot(x='LandMaxTemperature', y='LandAverageTem
```

```
perature', data=Temp)
fig.axis(ymin=0, ymax=800000)
xt = plt.xticks(rotation = 45)
```

OBSERVATION:

Here boxplot is plotted between LandMaxTemperature on x axis and LandAverageTemparature on y-axis

In [317]:

```
Temp['LandAverageTemperature'].unique()
```

Out[317]:

```
array([ 3.034,  3.083,  5.626, ..., 10.801,  7.433,  5.518])
```

We can observe the array of all the LandAverageTemparature that has unique values

In [318]:

```
Temp['LandAverageTemperature'].nunique()
```

Out[318]:

```
2839
```

OBSERVATION:

Here we can observe that nearly 2839 LandAverageTemparature had unique values

In [319]:

```
Temp['LandAverageTemperature'].value_counts()
```

Out[319]:

```
13.765    4
13.293    4
3.099     3
2.039     3
11.097    3
         ..
8.526     1
5.160     1
2.613     1
2.156     1
5.518     1
Name: LandAverageTemperature, Length: 2839, dtype: int64
```

OBSERVATION:

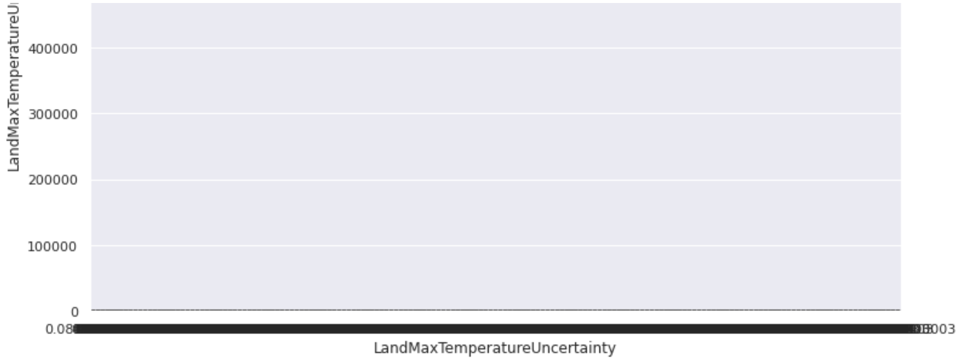
It gives the value counts of Land average temperature in Temp

In [321]:

```
f, ax = plt.subplots(figsize = (12,8))
fig = sns.boxplot(x='LandMaxTemperatureUncertainty', y='LandMaxTemperatureUncertainty', data=Temp
)
fig.axis(ymin=0, ymax=800000)
```

Out[321]:

```
(-0.5, 840.5, 0.0, 800000.0)
```



In [322]:

```
Temp['LandMaxTemperature'].value_counts()
```

Out[322]:

```
20.037    3
8.555     3
19.987    3
17.713    3
10.781    3
         ..
19.539    1
20.058    1
19.287    1
17.146    1
12.892    1
Name: LandMaxTemperature, Length: 1814, dtype: int64
```

DATA CLEANING

Dealing with missing values

In [323]:

```
Temp.columns
```

Out[323]:

```
Index(['dt', 'LandAverageTemperature', 'LandAverageTemperatu
reUncertainty',
       'LandMaxTemperature', 'LandMaxTemperatureUncertainty'
,
       'LandMinTemperature', 'LandMinTemperatureUncertainty'
,
       'LandAndOceanAverageTemperature',
       'LandAndOceanAverageTemperatureUncertainty'],
```

```
      dtype='object')
```

After observing the data, remove the data not required for the analysis and keep only the relevant data. The column 'LandAndOceanAverageTemperature' gives information about the overall earth temperature.

In [324]:

```
Temp.isnull().sum()
```

Out[324]:

```
dt                                              0
LandAverageTemperature                         12
LandAverageTemperatureUncertainty              12
LandMaxTemperature                           1200
LandMaxTemperatureUncertainty                1200
LandMinTemperature                           1200
LandMinTemperatureUncertainty                1200
LandAndOceanAverageTemperature               1200
LandAndOceanAverageTemperatureUncertainty    1200
dtype: int64
```

Here we find sum of all the null values in temp data

In [325]:

```
Temp[Temp['LandAverageTemperature'].isnull()]
```

Out[325]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxT
10	**1750-11-01**	**NaN**	**NaN**	
16	**1751-05-01**	**NaN**	**NaN**	
18	**1751-07-01**	**NaN**	**NaN**	
21	**1751-10-01**	**NaN**	**NaN**	
22	**1751-11-01**	**NaN**	**NaN**	
23	**1751-12-01**	**NaN**	**NaN**	

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxT
25	1752-02-01	NaN	NaN	
28	1752-05-01	NaN	NaN	
29	1752-06-01	NaN	NaN	
30	1752-07-01	NaN	NaN	
31	1752-08-01	NaN	NaN	
32	1752-09-01	NaN	NaN	

In [326]:

```
Temp. notnull().head
```

Out[326]:

```
<bound method NDFrame.head of          dt  LandAverageTempera
ture  LandAverageTemperatureUncertainty  \
0      True                        True
True
1      True                        True
True
2      True                        True
True
3      True                        True
True
4      True                        True
True
...     ...                         ...
...
3187   True                        True
True
3188   True                        True
True
3189   True                        True
True
3190   True                        True
True
3191   True                        True
True

      LandMaxTemperature  LandMaxTemperatureUncertainty  Lan
dMinTemperature  \
0                  False                          False
```

```
False
1                  False                          False
False
2                  False                          False
False
3                  False                          False
False
4                  False                          False
False
...                  ...                            ...
...
3187                True                           True
True
3188                True                           True
True
3189                True                           True
True
3190                True                           True
True
3191                True                           True
True

      LandMinTemperatureUncertainty  LandAndOceanAverageTemp
erature  \
0                             False
False
1                             False
False
2                             False
False
3                             False
False
4                             False
False
...                             ...
...
3187                           True
True
3188                           True
True
3189                           True
True
3190                           True
True
3191                           True
True

      LandAndOceanAverageTemperatureUncertainty
0                                         False
1                                         False
```

```
2                                                        False
3                                                        False
4                                                        False
...                                                        ...
3187                                                      True
3188                                                      True
3189                                                      True
3190                                                      True
3191                                                      True

[3192 rows x 9 columns]>
```

In [327]:

```python
Temp.shape
```

Out[327]:

```
(3192, 9)
```

In [328]:

```python
Temp.dropna(how='all').shape
```

Out[328]:

```
(3192, 9)
```

In [329]:

```python
Temp.dropna(how='any').shape
```

Out[329]:

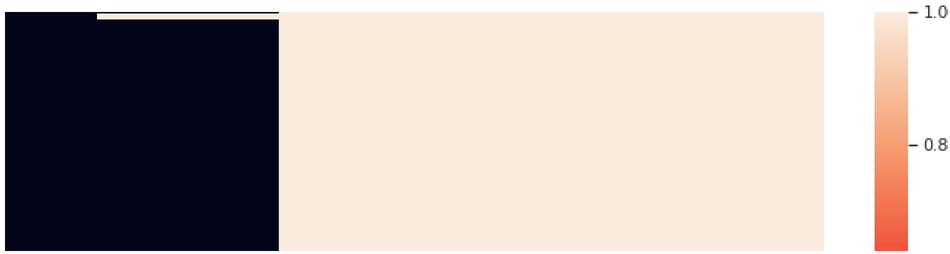
```
(1992, 9)
```

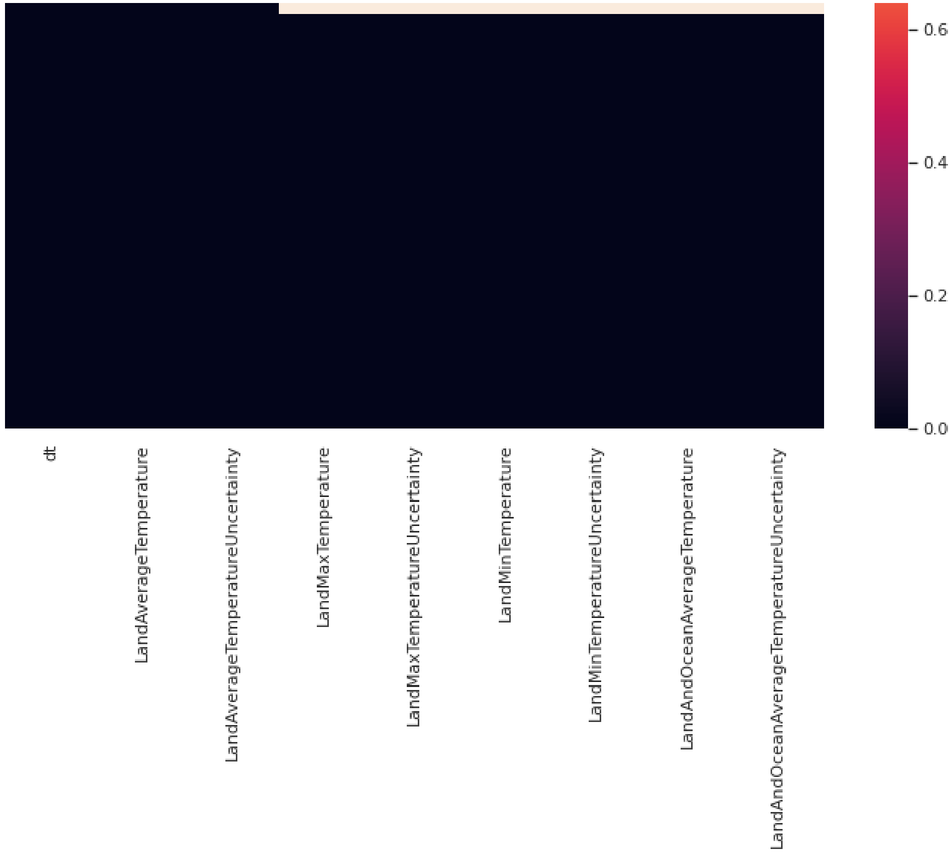
In [330]:

```python
sns. heatmap(Temp.isnull(), yticklabels=False)
```

Out[330]:

```
<AxesSubplot:>
```





MACHINE LEARNING MODELS

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

In [ ]:

```
X_Temp= Temp_1.drop('LandAverageTemperature',axis=1)
X_Temp= Temp_1.drop('LandAverageTemperature', axis = 1)
X_Temp
```

In [ ]:

```
X = X_Temp.drop('dt',axis=1)
X.head()
```

In [ ]:

```
y=X_Temp['LandAverageTemperatureUncertainty']
y.head()
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,tes
t_size=0.3, random_state=42)
```

In [ ]:

```
len(X_train),len(X_test)
```

LOGISTIC REGRESSION MODEL

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

In [ ]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train, y_train)
X_test = scaler.transform(X_test)
```

It is easy to implement and train a model using logistic regression

In [ ]:

```
from sklearn.linear_model import LogisticRegression
mod = LogisticRegression()
mod.fit(X_train, y_train)
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
mod = LogisticRegression()
mod.fit(X_train, y_train)
```

Logistic Regression uses a more complex cost function than Linear Regression, this cost function is called the 'Sigmoid function' or also known as the 'logistic function' instead of a linear function.

In [ ]:

```
from sklearn.metrics import confusion_matrix, classification
_report
y_pred_mod = mod.predict(X_test)
cf_matrix = confusion_matrix(y_test,y_pred_mod)
cf_matrixLogisticRegressionScore = mod.score(X_test, y_test
)
```

In [ ]:

```
sns.heatmap(cf_matrix, annot=True, cmap='inferno_r')
plt.title('Confusion Matrix for Logistic Regression',fontsiz
e=12, y=1.06)
```

For this purpose, a linear regression algorithm will help them decide. Plotting a regression line by considering the date as the independent variable, and the Land average temperature increase as the dependent variable will make their task easier

In [ ]:

```
from sklearn import metrics
print(metrics.classification_report(y_test, y_pred_mod))
```

In [ ]:

```
LogisticRegressionScore*100
```

In a regression problem, the output variable is a real continuous value

RANDOM FOREST MODEL

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:

```
RandomForestClassifierScore = rfc.score(X_test,y_test)
```

this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.

In [ ]:

```
y_pred_rfc = rfc.predict(X_test)
cf_matrix = confusion_matrix(y_test, y_pred_rfc)
cf_matrix
```

voting will be performed for every predicted result

At last, select the most voted prediction result as the final prediction result.

In [ ]:

```
sns.heatmap(cf_matrix, annot=True, cmap='viridis')
plt.title('Confusion Matrix for Random Forest Classifier',f
ontsize=12,y=1.06)print(metrics.classification_report(y_tes
t,y_pred_rfc))
```

In [ ]:

```
RandomForestClassifierScore*100
```

Random Forest algorithms maintains good accuracy even a large proportion of the data is missing.

DECISION TREE CLASSIFIER

Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

In [ ]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train,y_train)
```

root node present at the beginning of a decision tree from this node the population starts dividing according to various features.

In [ ]:

```
DecisionTreeClassifierScore = dtc.score(X_test,y_test)
```

In [ ]:

```
y_pred_dtc = dtc.predict(X_test)
cf_matrix = confusion_matrix(y_test,y_pred_dtc)
cf_matrix
```

It decrease uncertainty or disorders from the dataset

In [ ]:

```
sns.heatmap(cf_matrix, annot=True, cmap='tab20b_r')
plt.title("Confusion Matrix for Decision Tree Classifier",fo
ntsize=12, y=1.06)
```

In [ ]:

```
print(metrics.classification_report(y_test,y_pred_dtc))
```

Entropy is nothing but the uncertainty in our dataset or measure of disorder

In [ ]:

```
DecisionTreeClassifierScore*100
```

In [ ]:

```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier()
scores = cross_val_score(rf,X_train,y_train,cv=2,scoring='a
ccuracy')
print("scores :",scores)
print("mean :",scores.mean())
print("standard deviation :",scores.std())
```

In [ ]:

```
print("Accuracy obtained by LogisticRegressionModel : ",LogisticRegressionScore*100)
print("Accuracy obtained by RandomForestClassifierModel :",RandomForestClassifierScore*100)
print("Accuracy obtained by DecisionTreeClassifierModel :",DecisionTreeClassifierScore*100)
```

In [ ]:

```
Therefore the best Machine Learining Model obtained for our DataFrame is Decision Tree Classifier with more accuracy.
```