In [1]:
```
import pandas as pd
full_data = pd.read_csv('weatherAUS.csv')
full_data.head()
```

Out[1]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	101
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	100
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	101
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	101

5 rows × 23 columns

In [2]:
```
full_data.shape
full_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           145460 non-null  object 
 1   Location       145460 non-null  object 
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object 
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object 
 10  WindDir3pm     141232 non-null  object 
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object 
 22  RainTomorrow   142193 non-null  object 
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

In [3]:
```
full_data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
full_data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)
```

In [4]:
```
full_data=full_data.dropna()
```

In [5]:
```
full_data
```

Out[5]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
6049	2009-01-01	Cobar	17.9	35.2	0.0	12.0	12.3	SSW	48.0	ENE	...	20.0	13.0	
6050	2009-01-02	Cobar	18.4	28.9	0.0	14.8	13.0	S	37.0	SSE	...	30.0	8.0	
6052	2009-01-04	Cobar	19.4	37.6	0.0	10.8	10.6	NNE	46.0	NNE	...	42.0	22.0	
6053	2009-01-05	Cobar	21.9	38.4	0.0	11.4	12.2	WNW	31.0	WNW	...	37.0	22.0	
6054	2009-01-06	Cobar	24.2	41.0	0.0	11.2	8.4	WNW	35.0	NW	...	19.0	15.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142298	2017-06-20	Darwin	19.3	33.4	0.0	6.0	11.0	ENE	35.0	SE	...	63.0	32.0	
142299	2017-06-21	Darwin	21.2	32.6	0.0	7.6	8.6	E	37.0	SE	...	56.0	28.0	
142300	2017-06-22	Darwin	20.7	32.8	0.0	5.6	11.0	E	33.0	E	...	46.0	23.0	
142301	2017-06-23	Darwin	19.5	31.8	0.0	6.2	10.6	ESE	26.0	SE	...	62.0	58.0	
142302	2017-06-24	Darwin	20.2	31.7	0.0	5.6	10.7	ENE	30.0	ENE	...	73.0	32.0	

56420 rows × 23 columns

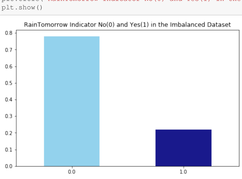
In [6]:
```
full_data.head()
full_data.tail()
```

Out[6]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
142298	2017-06-20	Darwin	19.3	33.4	0.0	6.0	11.0	ENE	35.0	SE	...	63.0	32.0	
142299	2017-06-21	Darwin	21.2	32.6	0.0	7.6	8.6	E	37.0	SE	...	56.0	28.0	
142300	2017-06-22	Darwin	20.7	32.8	0.0	5.6	11.0	E	33.0	E	...	46.0	23.0	
142301	2017-06-23	Darwin	19.5	31.8	0.0	6.2	10.6	ESE	26.0	SE	...	62.0	58.0	
142302	2017-06-24	Darwin	20.2	31.7	0.0	5.6	10.7	ENE	30.0	ENE	...	73.0	32.0	

5 rows × 23 columns

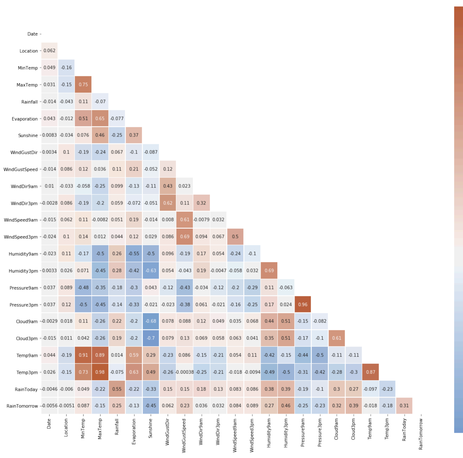
In [7]:
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,5))
full_data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
plt.show()
```



In [8]:
```
#Label Encoding
from sklearn.preprocessing import LabelEncoder
lencoders = {}
for col in full_data.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    full_data[col] = lencoders[col].fit_transform(full_data[col])
```

In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
corr = full_data.corr()
mask = np.triu(np.ones_like(corr, dtype=np.bool))
f, ax = plt.subplots(figsize=(20, 20))
cmap = sns.diverging_palette(250, 25, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None, center=0,square=True, annot=True, linewidths=.5, cbar_kws={"shrink": .9})
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x158ae27eb08>



In [10]:
```
# Standardizing data
from sklearn import preprocessing
r_scaler = preprocessing.MinMaxScaler()
r_scaler.fit(full_data)
modified_data = pd.DataFrame(r_scaler.transform(full_data), index=full_data.index, columns=full_data.columns)
```

In [11]:
```
# Feature Importance using Filter Method (Chi-Square)
from sklearn.feature_selection import SelectKBest, chi2
X = modified_data.loc[:,modified_data.columns!='RainTomorrow']
y = modified_data['RainTomorrow']
selector = SelectKBest(chi2, k=10)
selector.fit(X, y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Cloud9am', 'Cloud3pm', 'Temp3pm', 'RainToday'],
      dtype='object')
```

In [12]:
```
features=modified_data[['Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Cloud9am', 'Cloud3pm', 'Temp3pm', 'RainToday']]
```

In [13]:
```
target=modified_data['RainTomorrow']
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=12345)

# Normalize Features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100,random_state=0)
classifier.fit(X_train,y_train)
classifier.score(X_train,y_train)
```

Out[15]: 0.9999527354366065

In [16]:
```
y_pred = np.array(classifier.predict(X_test),dtype=int)
y_test = np.array(y_test,dtype=int)
```

In [17]:
```
print(y_pred)
print(y_test)
```

```
[0 1 0 ... 0 0 0]
[0 1 0 ... 1 0 0]
```

In [18]:
```
y_pred = y_pred.reshape(-1,1)
y_test = y_test.reshape(-1,1)
df = np.concatenate((y_test,y_pred),axis=1)
dataframe = pd.DataFrame(df,columns=['Rain on Tommorrow','Predition of Rain'])
print(dataframe)
print(type(y_pred))
print(type(y_test))
```

```
       Rain on Tommorrow  Predition of Rain
0                      0                  0
1                      1                  1
2                      0                  0
3                      0                  0
4                      1                  1
...                  ...                ...
14100                  0                  0
14101                  1                  0
14102                  1                  0
14103                  0                  0
14104                  0                  0

[14105 rows x 2 columns]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [19]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[19]: 0.8537398085785183

In [ ]: