Project 2 - Pecan Crop Decision Tree Analysis

In [1]:
```
## Load necessary Libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image
from dtreeviz import dtreeviz

## Hide warnings
import warnings
warnings.filterwarnings('ignore')

## Show entire dataframes
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Processing the Raw Data into a Single Dataframe

Individual Dataframes

Temperature Change

In [2]:
```
## Import data from csv into dataframe and then display dataframe
df1 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/Climate Change Indicators.csv')
climate = pd.DataFrame(df1)
climate.head()
```

Out[2]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Months Code	Months	Year Code	Year	Unit	Value	Flag	Flag Description
0	ET	Temperature change on land	840	United States of America	7271	Temperature change	7001	January	1961	1961	°c	0.950	E	Estimated value
1	ET	Temperature change on land	840	United States of America	7271	Temperature change	7001	January	1962	1962	°c	-0.772	E	Estimated value
2	ET	Temperature change on land	840	United States of America	7271	Temperature change	7001	January	1963	1963	°c	-1.184	E	Estimated value
3	ET	Temperature change on land	840	United States of America	7271	Temperature change	7001	January	1964	1964	°c	0.915	E	Estimated value
4	ET	Temperature change on land	840	United States of America	7271	Temperature change	7001	January	1965	1965	°c	0.753	E	Estimated value

In [3]:
```
## Extract desired columns into new dataframe with meaningful column names
tempchange = climate[['Year', 'Value']]
tempchange = tempchange.rename(columns={'Value': 'TempChangeC'})
tempchange.head()
```

Out[3]:

	Year	TempChangeC
0	1961	0.950
1	1962	-0.772
2	1963	-1.184
3	1964	0.915
4	1965	0.753

Pecan Production

In [4]:
```
## Import data from csv into dataframe and then display dataframe
df2 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/Crops and Livestock Products - Production Data.csv')
crops = pd.DataFrame(df2)
crops.head()
```

Out[4]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	QCL	Crops and livestock products	840	United States of America	5312	Area harvested	1379.9	Other nuts (excluding wild edible nuts and gro...	1985	1985	ha	65000	E	Estimated value	NaN
1	QCL	Crops and livestock products	840	United States of America	5419	Yield	1379.9	Other nuts (excluding wild edible nuts and gro...	1985	1985	100 g/ha	19985	I	Imputed value	NaN
2	QCL	Crops and livestock products	840	United States of America	5312	Area harvested	1379.9	Other nuts (excluding wild edible nuts and gro...	1986	1986	ha	65000	E	Estimated value	NaN
3	QCL	Crops and livestock products	840	United States of America	5419	Yield	1379.9	Other nuts (excluding wild edible nuts and gro...	1986	1986	100 g/ha	22100	I	Imputed value	NaN
4	QCL	Crops and livestock products	840	United States of America	5312	Area harvested	1379.9	Other nuts (excluding wild edible nuts and gro...	1987	1987	ha	65000	E	Estimated value	NaN

In [5]:
```
## Extract rows pertaining to crop yield each year
cropyield = crops[crops['Element']=='Yield']
# Extract desired columns into new dataframe with meaningful column names
pecancropprod = cropyield[['Year','Value']]
pecancropprod= pecancropprod.rename(columns={'Value': 'Yield100GperHa'})
pecancropprod.head()
```

Out[5]:

	Year	Yield100GperHa
1	1985	19985
3	1986	22100
5	1987	21262
7	1988	24662
9	1989	21000

In [6]:
```
## Extract rows pertaining to area harvested each year
cropharvest = crops[crops['Element']=='Area harvested']
# Extract desired columns into new dataframe with meainingful columnn names
haharvested= cropharvest[['Year','Value']]
haharvested = haharvested.rename(columns={'Value': 'HectaresHarvested'})
haharvested.head()
```

Out[6]:

	Year	HectaresHarvested
0	1985	65000
2	1986	65000
4	1987	65000
6	1988	65000
8	1989	65000

Export Value

In [7]:
```
## Import data from csv into dataframe and then display dataframe
df3 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/Producer Price Trade data - Pecans (export value only).csv')
prodprice = pd.DataFrame(df3)
prodprice.head()
```

Out[7]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1961	1961	1000 USD	2905	A	Official figure	NaN
1	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1962	1962	1000 USD	4279	A	Official figure	NaN
2	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1963	1963	1000 USD	4679	A	Official figure	NaN
3	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1964	1964	1000 USD	5671	A	Official figure	NaN
4	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1965	1965	1000 USD	4247	A	Official figure	NaN

In [8]:

```
## Extract desired columns into new dataframe with meaningful column names
exportvalue = prodprice[['Year', 'Value']]
exportvalue = exportvalue.rename(columns={'Value': 'ExportValue1000USD'})
exportvalue.head()
```

Out[8]:

	Year	ExportValue1000USD
0	1961	2905
1	1962	4279
2	1963	4679
3	1964	5671
4	1965	4247

Import Quantity

In [9]:

```
## Import data from csv into dataframe and then display dataframe
df4 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/Producer Price Trade data - Pecans.csv')
imports = pd.DataFrame(df4)
imports.head()
```

Out[9]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	TCL	Crops and livestock products	840	United States of America	5610	Import Quantity	1379.9	Other nuts (excluding wild edible nuts and gro...	1961	1961	t	266.0	A	Official figure	NaN
1	TCL	Crops and livestock products	840	United States of America	5622	Import Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1961	1961	1000 USD	417.0	A	Official figure	NaN
2	TCL	Crops and livestock products	840	United States of America	5910	Export Quantity	1379.9	Other nuts (excluding wild edible nuts and gro...	1961	1961	t	2857.0	A	Official figure	NaN
3	TCL	Crops and livestock products	840	United States of America	5922	Export Value	1379.9	Other nuts (excluding wild edible nuts and gro...	1961	1961	1000 USD	2905.0	A	Official figure	NaN
4	TCL	Crops and livestock products	840	United States of America	5610	Import Quantity	1379.9	Other nuts (excluding wild edible nuts and gro...	1962	1962	t	791.0	A	Official figure	NaN

In [10]:
```
## Extract desired columns into new dataframe with meaningful column names
importquant = imports[imports['Element']=='Import Quantity']
# Extract desired columns into new dataframe with meainingful columnn names
importquant = importquant[['Year','Value']]
importquant = importquant.rename(columns={'Value': 'ImportTons'})
importquant.head()
```

Out[10]:

	Year	ImportTons
0	1961	266.0
4	1962	791.0
8	1963	608.0
12	1964	497.0
16	1965	371.0

Producer Price

In [11]:
```
## Import data from csv into dataframe and then display dataframe
df5= pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/Producer Prices - Annual Value - Pecans.csv')
```

```
producerprice = pd.DataFrame(df5)
producerprice.head()
```

Out[11]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Months Code	Months	Unit	Value	Flag	Flag Description
0	PP	Producer Prices	840	United States of America	5532	Producer Price (USD/tonne)	1379.9	Other nuts (excluding wild edible nuts and gro...	1991	1991	7021	Annual value	USD	2293	A	Official figure
1	PP	Producer Prices	840	United States of America	5532	Producer Price (USD/tonne)	1379.9	Other nuts (excluding wild edible nuts and gro...	1992	1992	7021	Annual value	USD	3197	A	Official figure
2	PP	Producer Prices	840	United States of America	5532	Producer Price (USD/tonne)	1379.9	Other nuts (excluding wild edible nuts and gro...	1993	1993	7021	Annual value	USD	1292	A	Official figure
3	PP	Producer Prices	840	United States of America	5532	Producer Price (USD/tonne)	1379.9	Other nuts (excluding wild edible nuts and gro...	1994	1994	7021	Annual value	USD	2293	A	Official figure
4	PP	Producer Prices	840	United States of America	5532	Producer Price (USD/tonne)	1379.9	Other nuts (excluding wild edible nuts and gro...	1995	1995	7021	Annual value	USD	2227	A	Official figure

In [12]:

```
## Extract desired columns into new dataframe with meaningful column names
producerprice = producerprice[['Year','Value']]
producerprice = producerprice.rename(columns={'Value': 'ProdUSDperTonne'})
producerprice.head()
```

Out[12]:

	Year	ProdUSDperTonne
0	1991	2293
1	1992	3197
2	1993	1292
3	1994	2293
4	1995	2227

Water Stress

In [13]:
```
## Import data from csv into dataframe and then display dataframe
df6= pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/SDG Indicators Level of Water of Stress.csv')
water = pd.DataFrame(df6)
water.head()
```

Out[13]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (SDG)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note
0	SDGB	SDG Indicators	840	United States of America	6121	Value	ER_H2O_STRESS_AGR	6.4.2 Level of water stress (Agriculture (ISIC...	2000	2000	%	10.54	A	Official figure	Global monitoring data | Food and Agriculture ...
1	SDGB	SDG Indicators	840	United States of America	6121	Value	ER_H2O_STRESS_AGR	6.4.2 Level of water stress (Agriculture (ISIC...	2001	2001	%	10.51	A	Official figure	Global monitoring data | Food and Agriculture ...
2	SDGB	SDG Indicators	840	United States of America	6121	Value	ER_H2O_STRESS_AGR	6.4.2 Level of water stress (Agriculture (ISIC...	2002	2002	%	10.47	A	Official figure	Global monitoring data | Food and Agriculture ...
3	SDGB	SDG Indicators	840	United States of America	6121	Value	ER_H2O_STRESS_AGR	6.4.2 Level of water stress (Agriculture (ISIC...	2003	2003	%	10.43	A	Official figure	Global monitoring data | Food and Agriculture ...
4	SDGB	SDG Indicators	840	United States of America	6121	Value	ER_H2O_STRESS_AGR	6.4.2 Level of water stress (Agriculture (ISIC...	2004	2004	%	10.39	A	Official figure	Global monitoring data | Food and Agriculture ...

In [14]:
```
# Select rows of dataframe that pertain to agriculutral water stress
waterstress= water[water['Item']=='6.4.2 Level of water stress (Agriculture (ISIC4 A01 A0210 A0322))']
## Extract desired columns into new dataframe with meaningful column names
waterstress = waterstress[['Year','Value']]
waterstress = waterstress.rename(columns={'Value': 'WaterStressPerct'})
waterstress.head()
```

Out[14]:

	Year	WaterStressPerct
0	2000	10.54
1	2001	10.51
2	2002	10.47
3	2003	10.43
4	2004	10.39

Combining the different frames together

Dataframes to combine:

- tempchange
- pecancropprod
- haharvested
- exportvalue
- importquant
- producerprice
- waterstress

In [15]:

```
## Combine separate dataframes into one and fill missing values with blanks
mergeddata = tempchange.merge(pecancropprod, on='Year', how='left').fillna('')
mergeddata = mergeddata.merge(haharvested, on='Year', how='left').fillna('')
mergeddata = mergeddata.merge(exportvalue, on='Year', how='left').fillna('')
mergeddata = mergeddata.merge(importquant, on='Year', how='left').fillna('')
mergeddata = mergeddata.merge(producerprice, on='Year', how='left').fillna('')
mergeddata = mergeddata.merge(waterstress, on='Year', how='left').fillna('')
mergeddata = mergeddata.groupby(['Year']).first().reset_index()
mergeddata.head()
```

Out[15]:

	Year	TempChangeC	Yield100GperHa	HectaresHarvested	ExportValue1000USD	ImportTons	ProdUSDperTonne	WaterStressPerct
0	1961	0.950			2905.0	266.0		
1	1962	-0.772			4279.0	791.0		
2	1963	-1.184			4679.0	608.0		
3	1964	0.915			5671.0	497.0		
4	1965	0.753			4247.0	371.0		

In [16]:

```
mergeddata.dtypes
```

Out[16]:

```
Year                    int64
TempChangeC           float64
Yield100GperHa         object
HectaresHarvested      object
ExportValue1000USD     object
ImportTons             object
ProdUSDperTonne        object
WaterStressPerct       object
dtype: object
```

After taking a look at the mergedata, and writing it to a csv file, the only years that had data that would be useful for this project are the years 2000-2017. Therefore, I will center the focus of these particular years, and create a new dataframe from this range.

In [17]:

```
## Extract rows from mergeddata to show only years 2000-2017
newdf = mergeddata[mergeddata['Year']<=2017]
newdf = newdf[mergeddata['Year']>=2000]
## make sure variables are numeric
newdf['Yield100GperHa'] = pd.to_numeric(newdf['Yield100GperHa'])
newdf['HectaresHarvested'] = pd.to_numeric(newdf['HectaresHarvested'])
newdf['ExportValue1000USD'] = pd.to_numeric(newdf['ExportValue1000USD'])
newdf['ImportTons'] = pd.to_numeric(newdf['ImportTons'])
newdf['ProdUSDperTonne'] = pd.to_numeric(newdf['ProdUSDperTonne'])
newdf['WaterStressPerct'] = pd.to_numeric(newdf['WaterStressPerct'])
newdf.head()
```

Out[17]:

	Year	TempChangeC	Yield100GperHa	HectaresHarvested	ExportValue1000USD	ImportTons	ProdUSDperTonne	WaterStressPerct
39	2000	1.543	26189.0	45000.0	107315.0	39130.61	2513.0	10.54
40	2001	2.055	25941.0	69000.0	82715.0	27963.00	1310.0	10.51
41	2002	2.783	25628.0	40000.0	86719.0	36969.00	2105.0	10.47
42	2003	2.167	26215.0	58000.0	89797.0	44374.00	2169.0	10.43
43	2004	0.116	25871.0	42500.0	124934.0	58973.00	3880.0	10.39

In [18]:

```
## Write new datafram to csv file
newdf.to_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project 2/PecanProjectData.csv', index = False)
```

In [19]:

```
## print out descriptive statistics for each variable in new dataframe.
newdf.describe()
```

Out[19]:

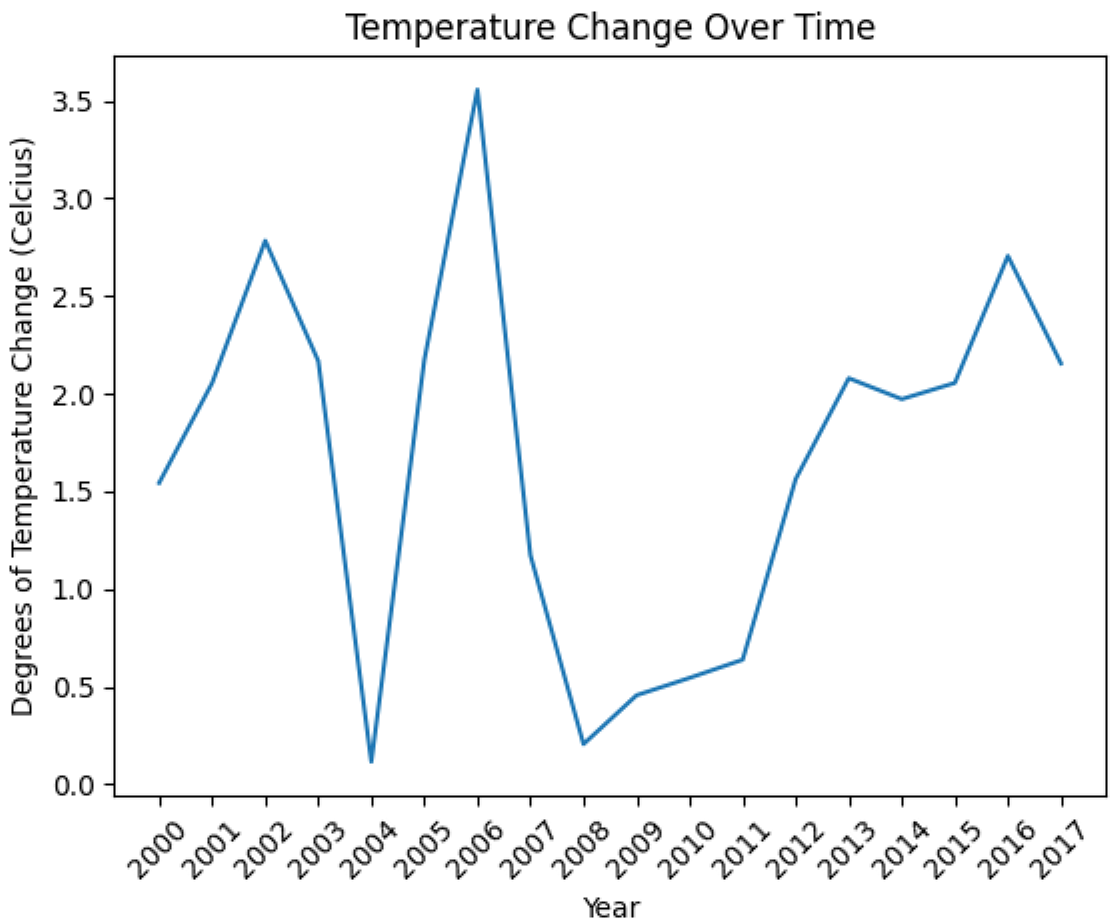
	Year	TempChangeC	Yield100GperHa	HectaresHarvested	ExportValue1000USD	ImportTons	ProdUSDperTonne	WaterStressPerct
count	18.000000	18.000000	18.000000	18.000000	18.000000	18.000000	18.000000	18.000000
mean	2008.500000	1.663333	21799.222222	67647.555556	306843.722222	58364.738889	3606.111111	10.386667
std	5.338539	0.963926	5685.854000	37397.572197	203251.460021	15517.673908	1262.201992	0.481285
min	2000.000000	0.116000	7652.000000	38000.000000	82715.000000	27963.000000	1310.000000	9.570000
25%	2004.250000	0.771500	21007.500000	45175.000000	119432.000000	50603.750000	2623.250000	10.072500
50%	2008.500000	2.013500	24150.500000	58658.000000	258564.000000	56920.000000	3450.000000	10.410000
75%	2012.750000	2.164000	25731.500000	64204.250000	497792.000000	73602.947500	4717.750000	10.532500
max	2017.000000	3.556000	26215.000000	170700.000000	709596.000000	85363.460000	5710.000000	11.170000

Data Exploration

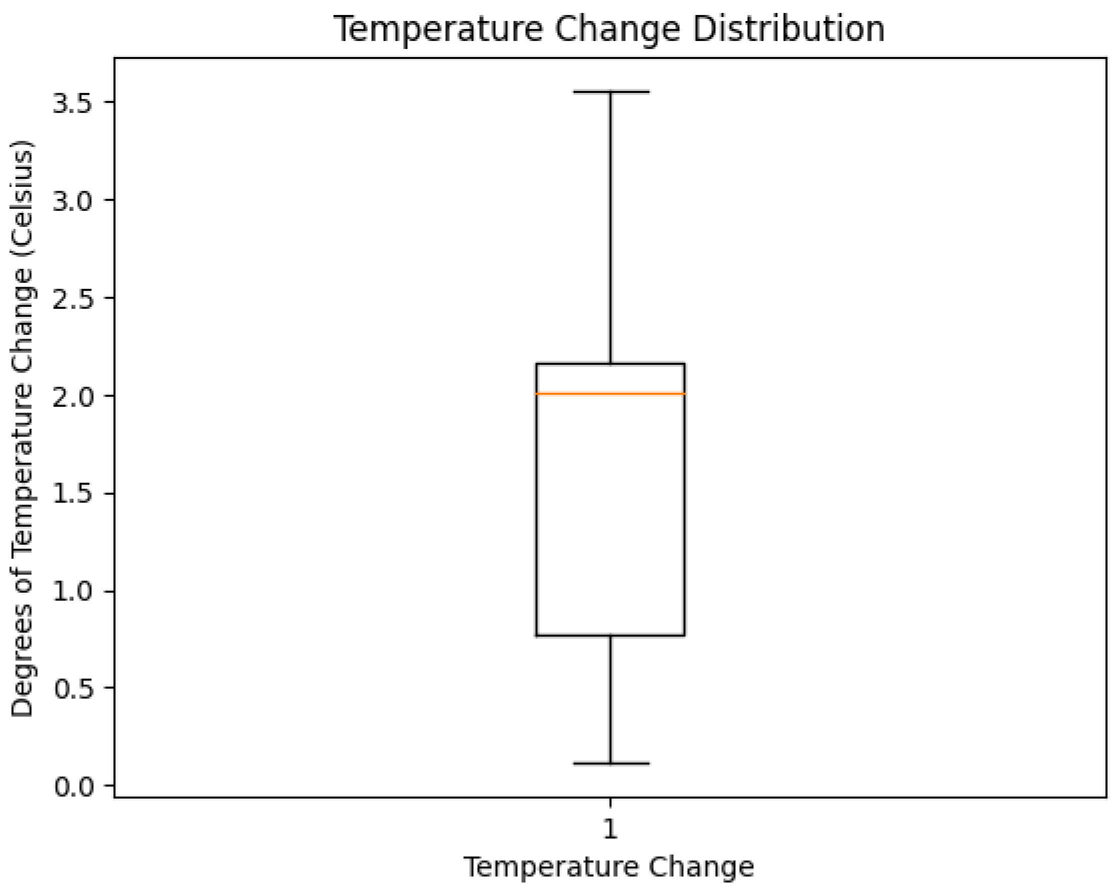
Trends

TempChangeC

In [20]:
```
## Line plot on TempChangeC column
plt.plot(newdf['Year'], newdf['TempChangeC'])
plt.xlabel('Year')
plt.ylabel('Degrees of Temperature Change (Celcius)')
plt.title('Temperature Change Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```



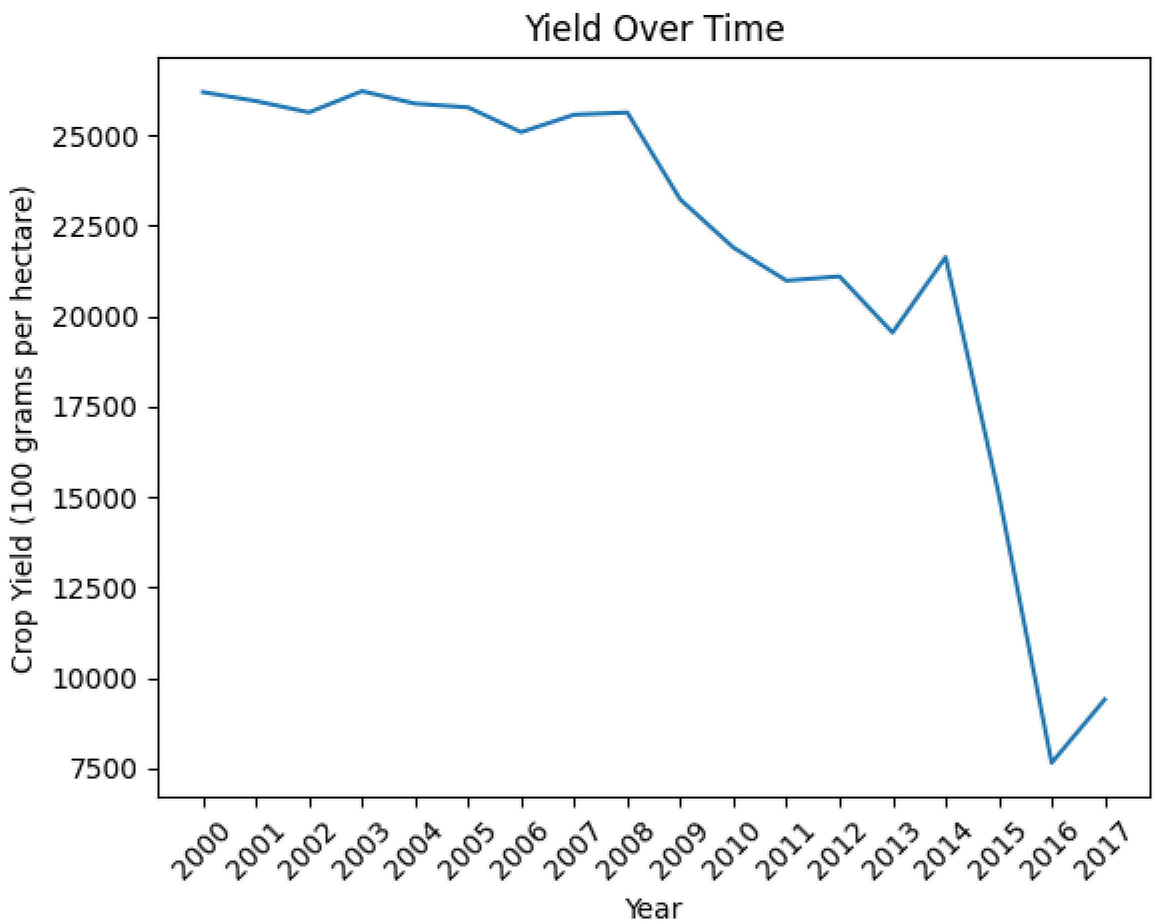
In [21]:
```
## Box plot on TempChangeC column
plt.boxplot(newdf['TempChangeC'])
plt.xlabel('Temperature Change')
plt.ylabel('Degrees of Temperature Change (Celsius)')
plt.title('Temperature Change Distribution')
plt.show()
```



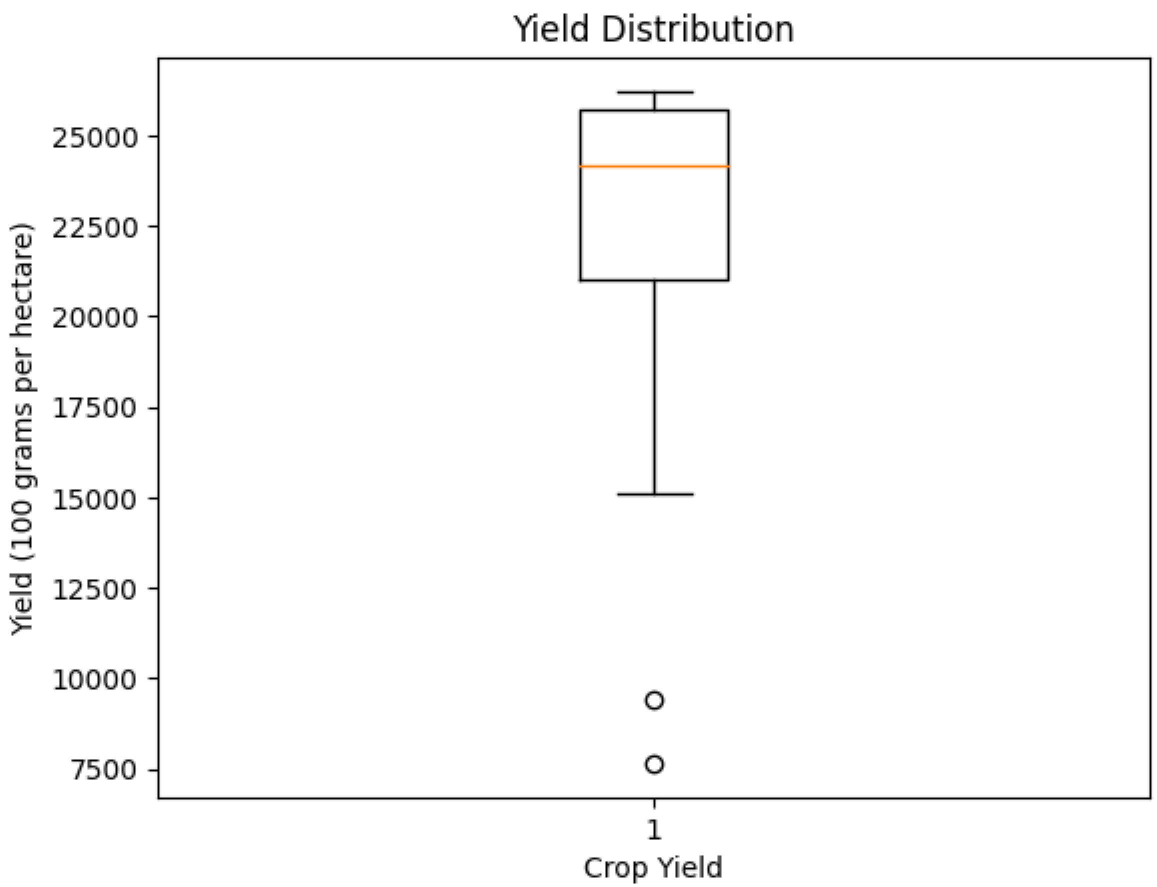
Yield100GperHa

In [22]:
```
## Line plot on Yield100GperHa Column
plt.plot(newdf['Year'], newdf['Yield100GperHa'])
plt.xlabel('Year')
plt.ylabel('Crop Yield (100 grams per hectare)')
```

```
plt.title('Yield Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```

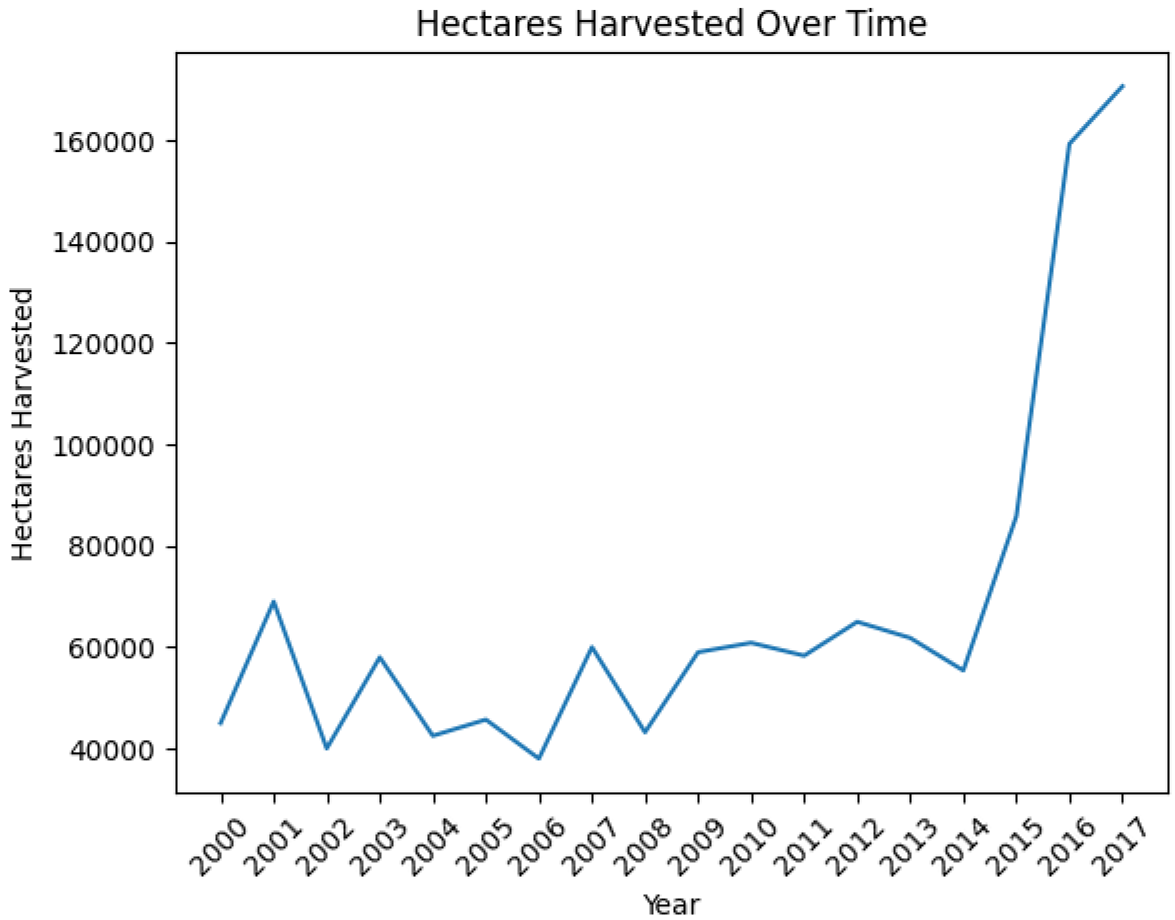


In [23]:
```
## Box plot on Yield100GperHa Column
plt.boxplot(newdf['Yield100GperHa'])
plt.xlabel('Crop Yield')
plt.ylabel('Yield (100 grams per hectare)')
plt.title('Yield Distribution')
plt.show()
```

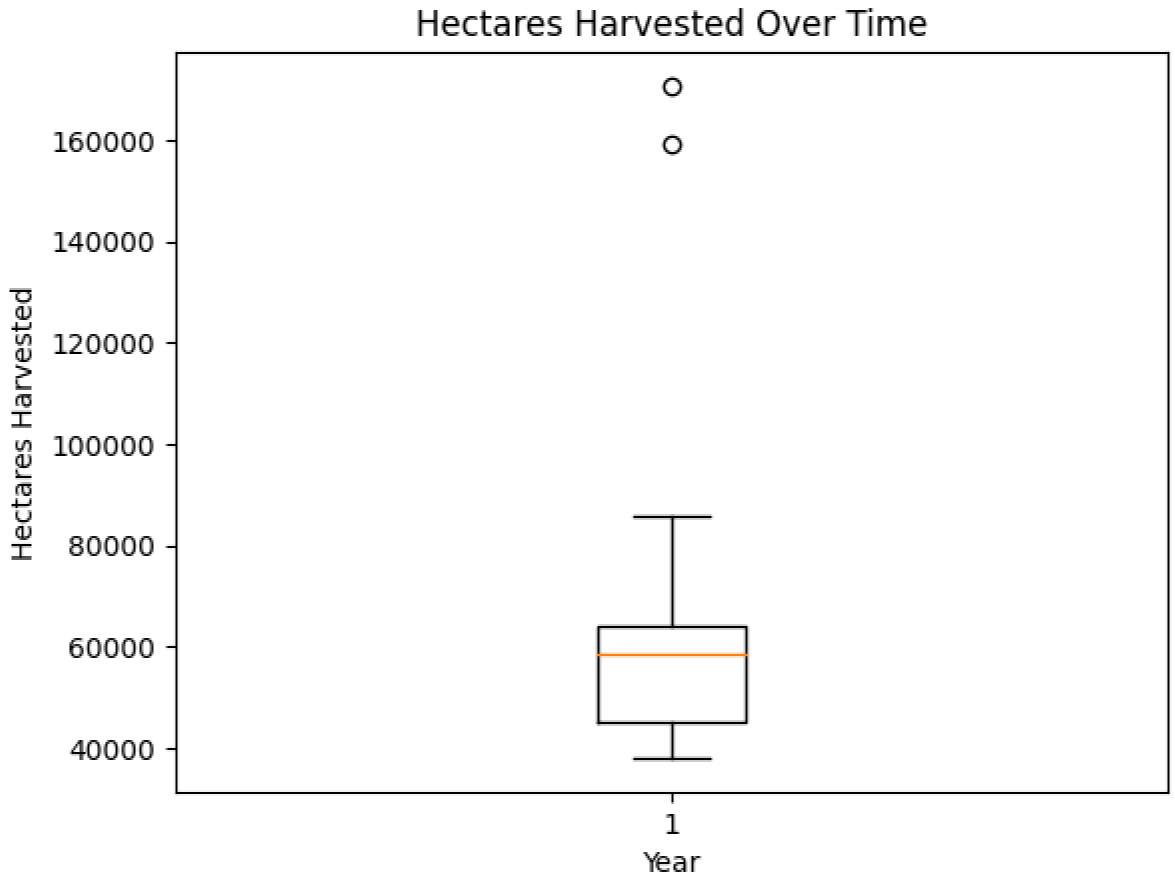


HectaresHarvested

In [24]:
```
## Line plot on HectaresHarvested column
plt.plot(newdf['Year'], newdf['HectaresHarvested'])
plt.xlabel('Year')
plt.ylabel('Hectares Harvested')
plt.title('Hectares Harvested Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```



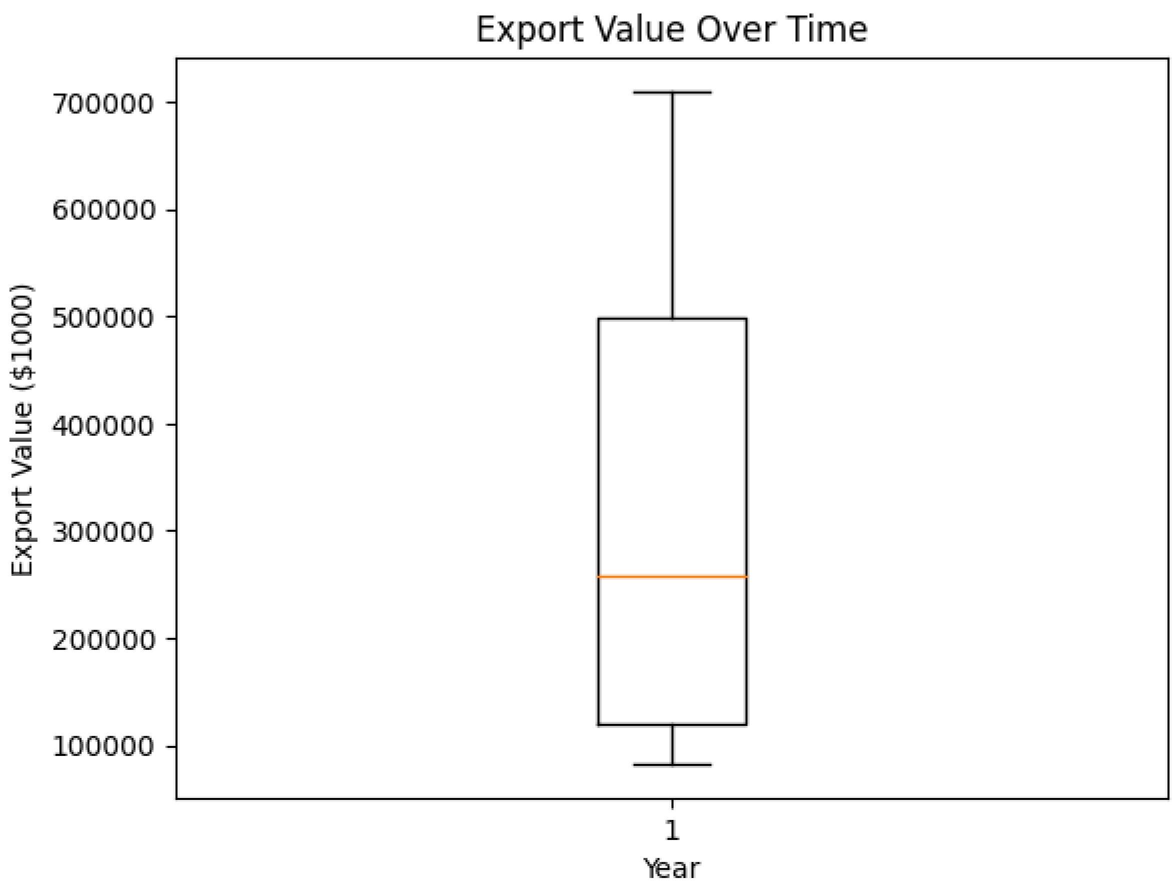
In [25]:
```
## Box plot on HectaresHarvested column
plt.boxplot(newdf['HectaresHarvested'])
plt.xlabel('Year')
plt.ylabel('Hectares Harvested')
plt.title('Hectares Harvested Over Time')
plt.show()
```



ExportValue1000USD

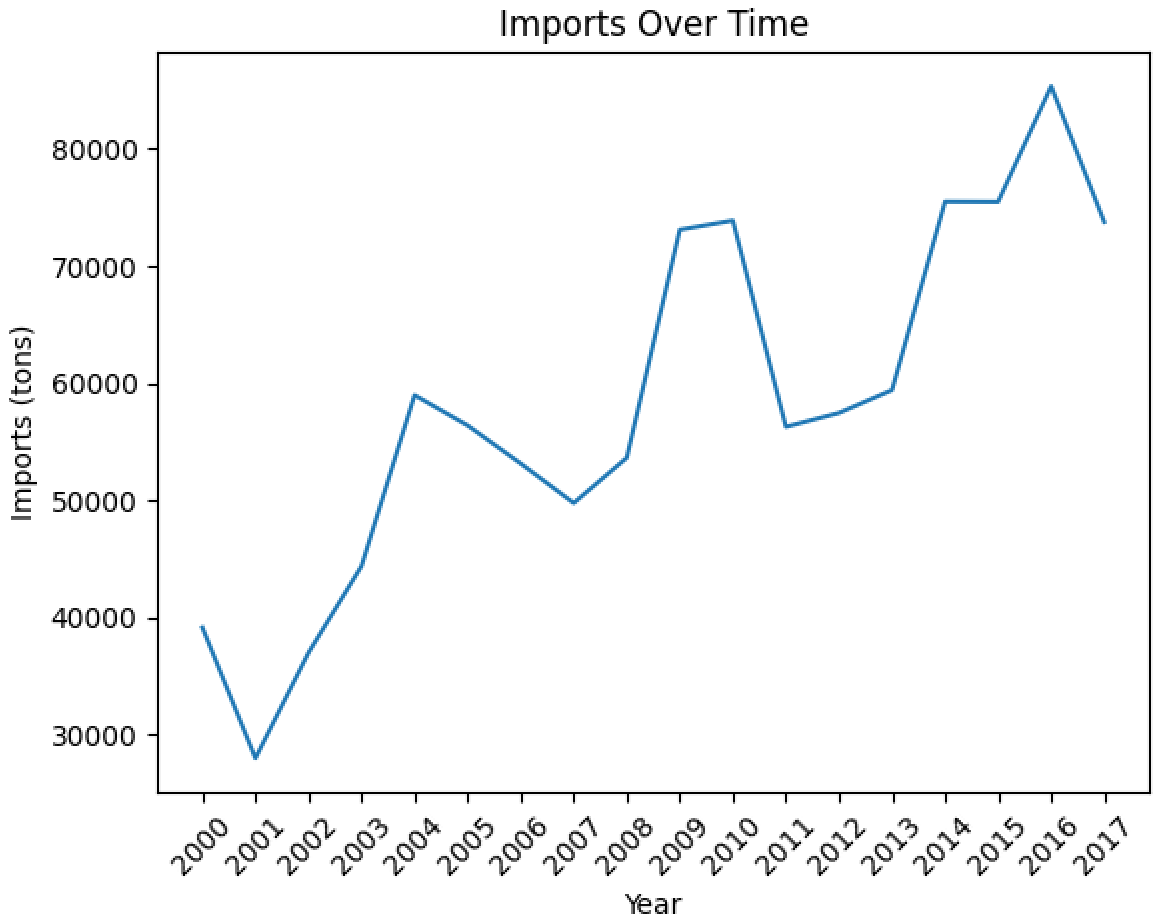
In [26]:
```
## Line plot on ExportValue1000USD column
plt.plot(newdf['Year'], newdf['ExportValue1000USD'])
plt.xlabel('Year')
plt.ylabel('Export Value ($1000)')
plt.title('Export Value Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```

In [27]:
```
## Box plot on ExportValue1000USD column
plt.boxplot(newdf['ExportValue1000USD'])
plt.xlabel('Year')
plt.ylabel('Export Value ($1000)')
plt.title('Export Value Over Time')
plt.show()
```

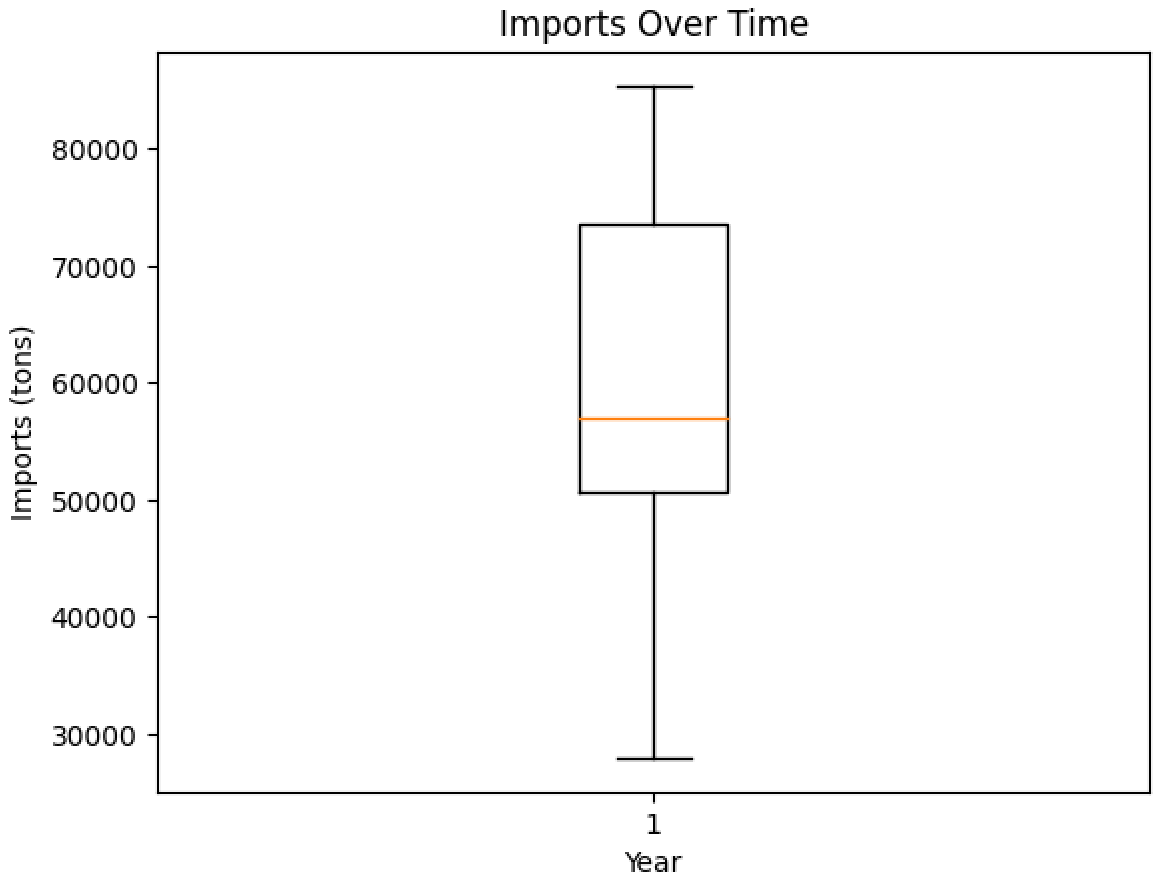


ImportTons

In [28]:
```
## Line plot on ImportTons column
plt.plot(newdf['Year'], newdf['ImportTons'])
plt.xlabel('Year')
plt.ylabel('Imports (tons)')
plt.title('Imports Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```

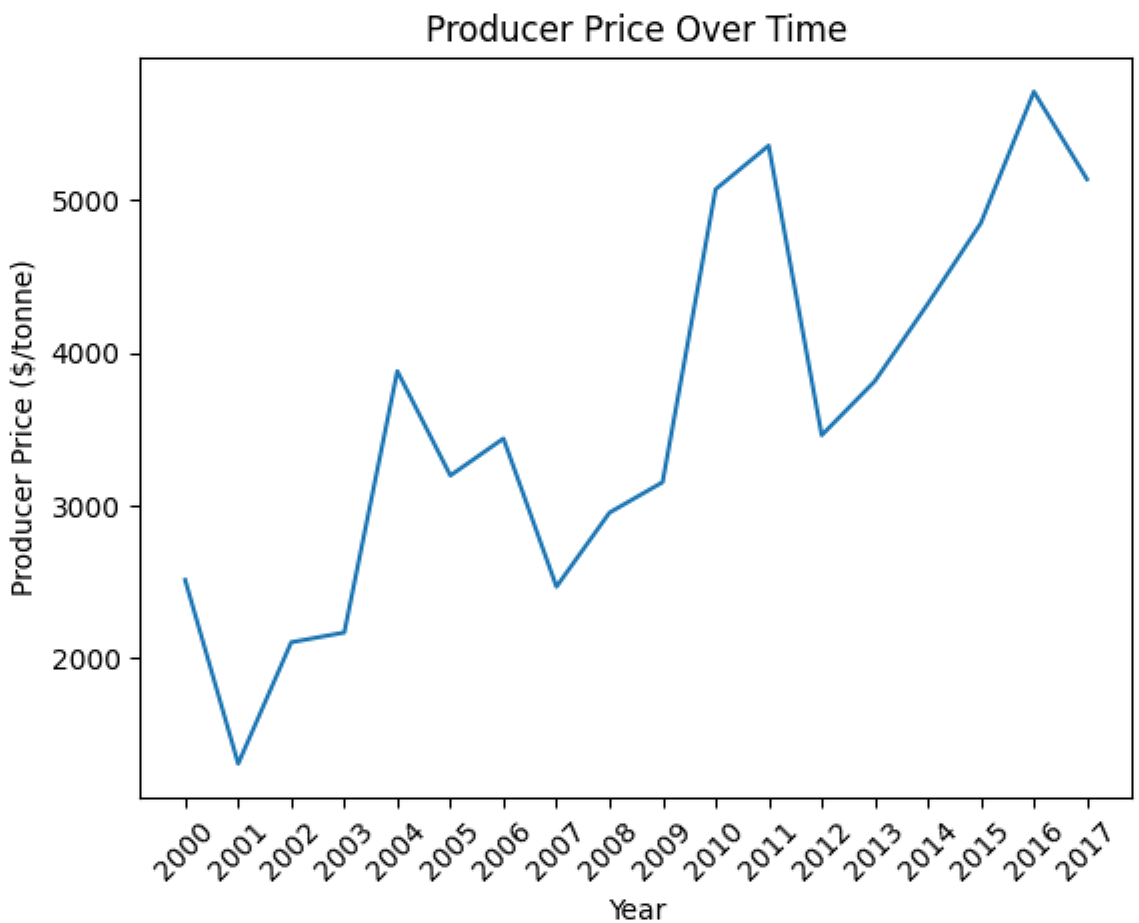


In [29]:
```
## Box plot on ImportTons column
plt.boxplot(newdf['ImportTons'])
plt.xlabel('Year')
plt.ylabel('Imports (tons)')
plt.title('Imports Over Time')
plt.show()
```

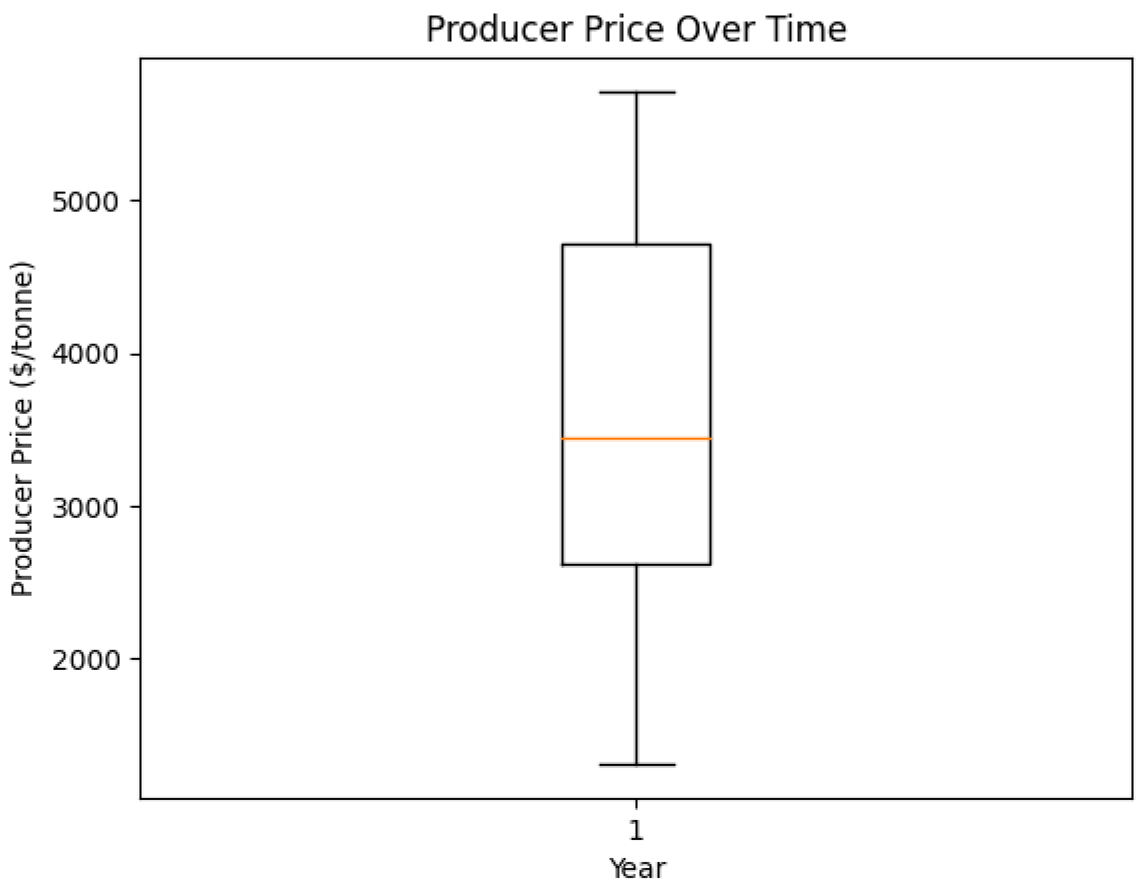


ProdUSDperTonne

In [30]:
```
## Line plot on ProdUSDperTonne column
plt.plot(newdf['Year'], newdf['ProdUSDperTonne'])
plt.xlabel('Year')
plt.ylabel('Producer Price ($/tonne)')
plt.title('Producer Price Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```

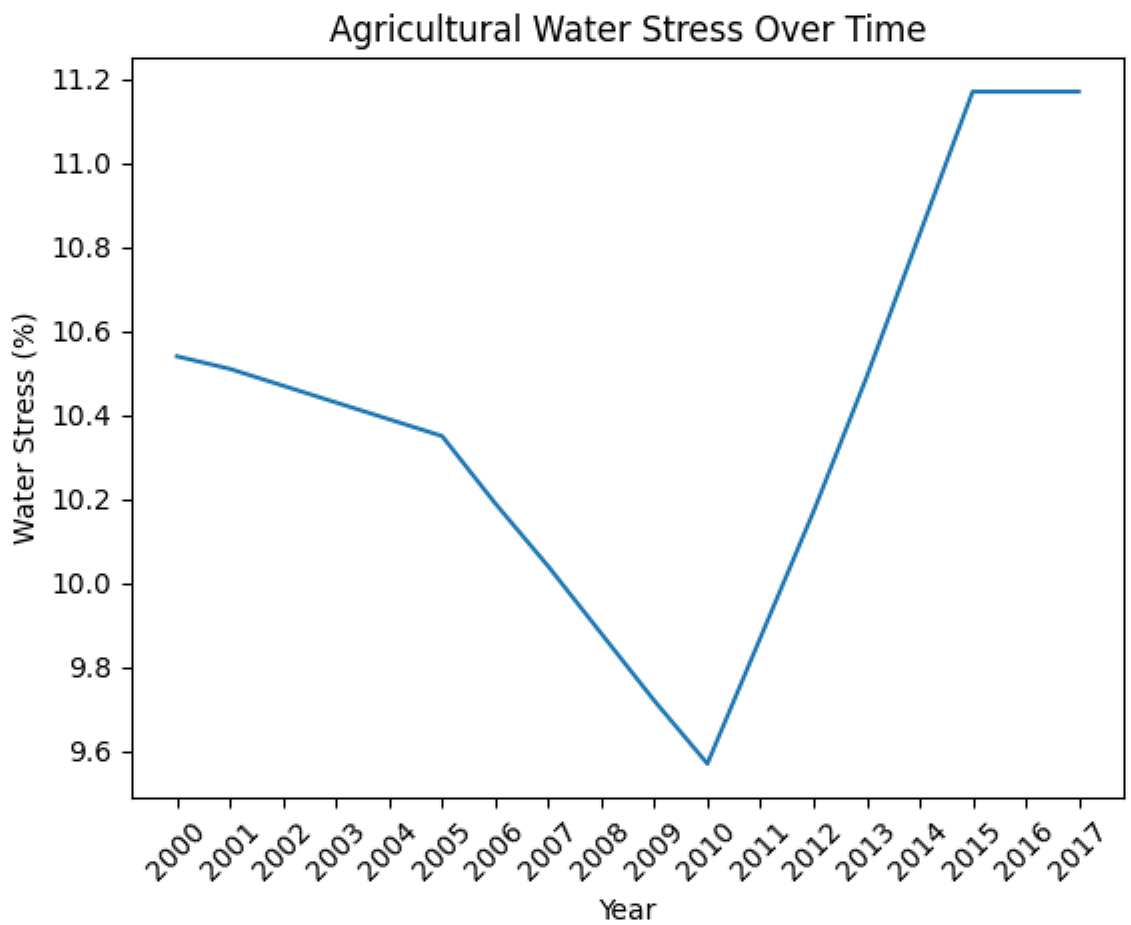


In [31]:
```
## Box plot on ProdUSDperTonne column
plt.boxplot(newdf['ProdUSDperTonne'])
plt.xlabel('Year')
plt.ylabel('Producer Price ($/tonne)')
plt.title('Producer Price Over Time')
plt.show()
```

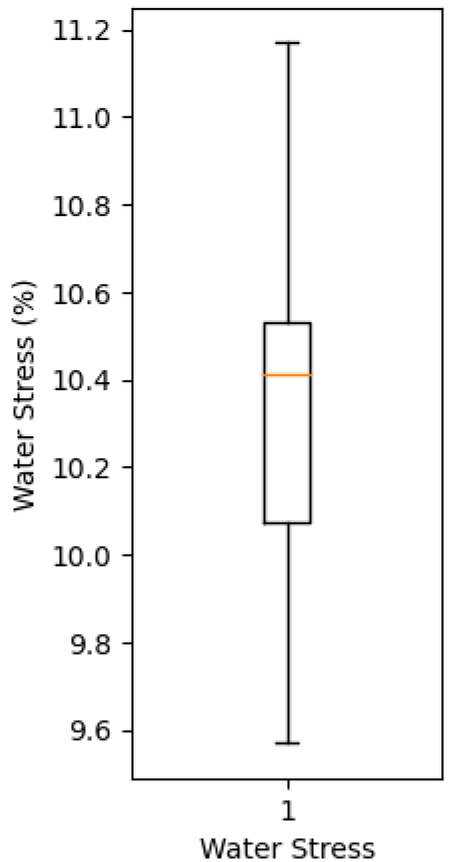


WaterStressPerct

In [32]:
```
## Line plot on WaterStressPerct column
plt.plot(newdf['Year'], newdf['WaterStressPerct'])
plt.xlabel('Year')
plt.ylabel('Water Stress (%)')
plt.title('Agricultural Water Stress Over Time')
plt.xticks(newdf['Year'].astype(int), rotation=45)
plt.show()
```



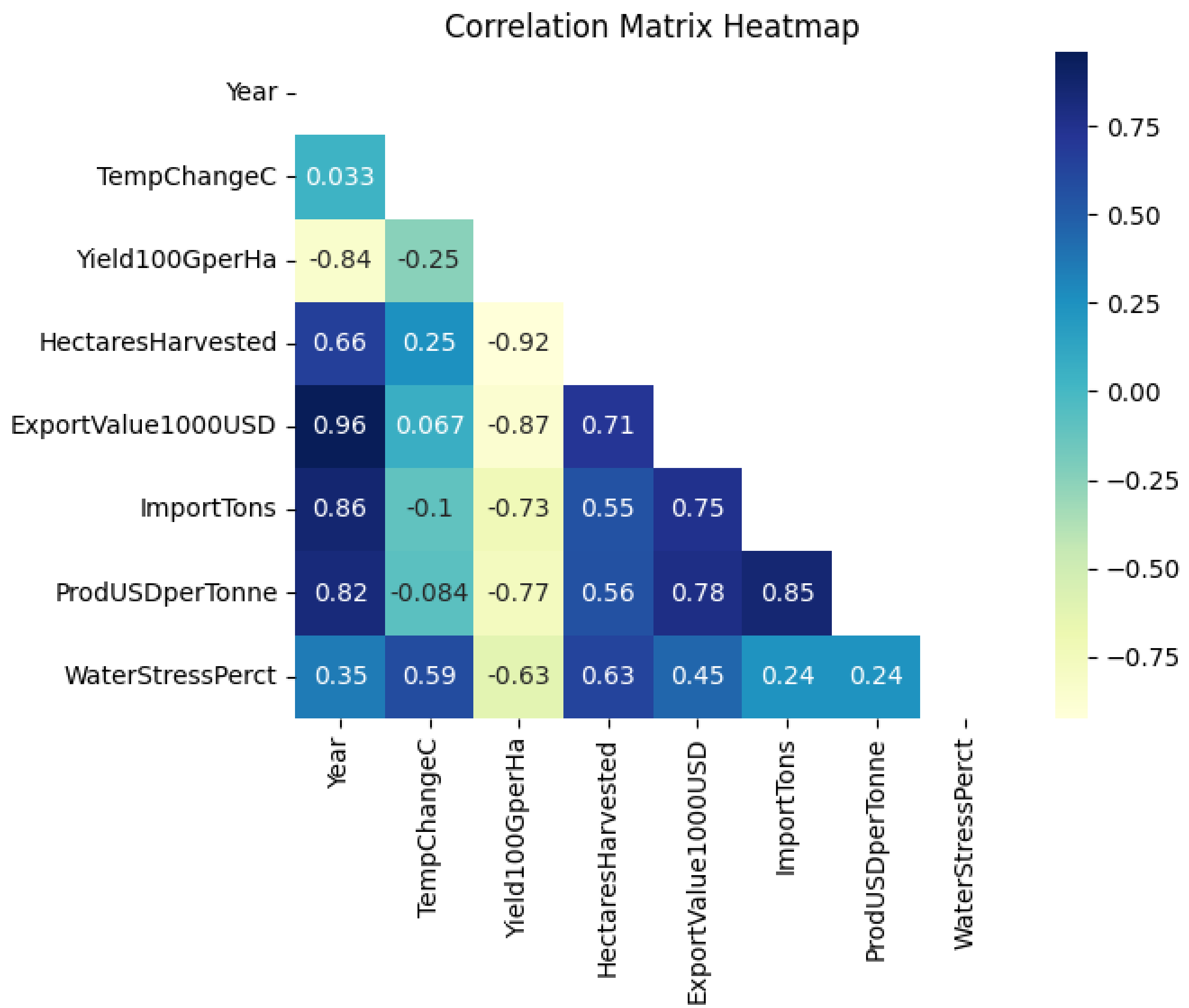
In [33]:
```
## Box plot on WaterStressPerct column
plt.figure(figsize = (2,5))
plt.boxplot(newdf['WaterStressPerct'])
plt.xlabel('Water Stress')
plt.ylabel('Water Stress (%)')
plt.show()
```



Correlation Matrixes

In [34]:
```
## Create triangular heat map of correlations between variables to see highly correlated ones.
matrix = newdf.corr()

## Define a mask to hide upper portion of heat map with mirrored values
mask = np.triu(np.ones_like(matrix, dtype=np.bool))

## Build and color heatmap
heatmap = sns.heatmap(matrix, mask=mask, annot=True, cmap="YlGnBu")
heatmap.set_title('Correlation Matrix Heatmap', fontdict={'fontsize':12})
```

Out[34]: `Text(0.5, 1.0, 'Correlation Matrix Heatmap')`



Train and Test Dataset

In [35]:
```
## Remove Year column from dataset
newdf = newdf.drop('Year', axis=1)

## Define threshold values to create categories for models
```

```
threshold_low = 9.0
threshold_medium = 10.0
threshold_high = 12.0
```

In [36]:
```
## Create categories based on thresholds
newdf['WaterStressCategory'] = pd.cut(newdf['WaterStressPerct'], bins=[-float('inf'), threshold_low, threshold_medium, threshold_high, float('inf')],
                                      labels=['Low', 'Medium', 'High', 'Very High'])

## Define features (X) and target (y)
X = newdf.drop(['WaterStressPerct', 'WaterStressCategory'], axis=1)
y = newdf['WaterStressCategory']

## Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Neural Network

In [37]:
```
## Standardize features for Neural Network
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

## Create Neural Network Model
neural_network = MLPClassifier(hidden_layer_sizes=(32, 32), activation='relu', solver='adam', max_iter=10, random_state=0)

## Train Neural Network
neural_network.fit(X_train_scaled, y_train)

## Make predictions
neural_network_train_predictions = neural_network.predict(X_train_scaled)
neural_network_test_predictions = neural_network.predict(X_test_scaled)

## Calculate and print accuracy
neural_network_train_accuracy = accuracy_score(y_train, neural_network_train_predictions)
neural_network_test_accuracy = accuracy_score(y_test, neural_network_test_predictions)

print("Neural Network Training Accuracy:", neural_network_train_accuracy)
print("Neural Network Testing Accuracy:", neural_network_test_accuracy)
```

```
Neural Network Training Accuracy: 0.8571428571428571
Neural Network Testing Accuracy: 1.0
```

In [38]:

```
## Print classification report
nn_report = classification_report(y_test, neural_network_test_predictions)
print("Classification Report for Neural Network:\n", nn_report)
```

```
Classification Report for Neural Network:
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         3
      Medium       1.00      1.00      1.00         1

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

Random Forest

In [39]:

```
## Create Random Forest and train
random_forest = RandomForestClassifier(n_estimators=100, random_state=0)
random_forest.fit(X_train_scaled, y_train)

## Make predictions
random_forest_train_predictions = random_forest.predict(X_train_scaled)
random_forest_test_predictions = random_forest.predict(X_test_scaled)

## Calculate and print accuracy
random_forest_train_accuracy = accuracy_score(y_train, random_forest_train_predictions)
random_forest_test_accuracy = accuracy_score(y_test, random_forest_test_predictions)

print("Random Forest Training Accuracy:", random_forest_train_accuracy)
print("Random Forest Testing Accuracy:", random_forest_test_accuracy)
```

```
Random Forest Training Accuracy: 1.0
Random Forest Testing Accuracy: 0.75
```

In [40]:

```
## Print classification report
rf_report = classification_report(y_test, random_forest_test_predictions)
print("Classification Report for Random Forest:\n", rf_report)
```

```
Classification Report for Random Forest:
               precision    recall  f1-score   support

        High       0.75      1.00      0.86         3
      Medium       0.00      0.00      0.00         1

    accuracy                           0.75         4
   macro avg       0.38      0.50      0.43         4
weighted avg       0.56      0.75      0.64         4
```

Decision Tree Classifier

In [41]:
```
## Create Decision Tree Classifier and train
decision_tree = DecisionTreeClassifier(random_state=0)
decision_tree.fit(X_train_scaled, y_train)

## Make predictions
decision_tree_train_predictions = decision_tree.predict(X_train_scaled)
decision_tree_test_predictions = decision_tree.predict(X_test_scaled)

## Calculate and print accuracy
decision_tree_train_accuracy = accuracy_score(y_train, decision_tree_train_predictions)
decision_tree_test_accuracy = accuracy_score(y_test, decision_tree_test_predictions)

print("Decision Tree Training Accuracy:", decision_tree_train_accuracy)
print("Decision Tree Testing Accuracy:", decision_tree_test_accuracy)
```

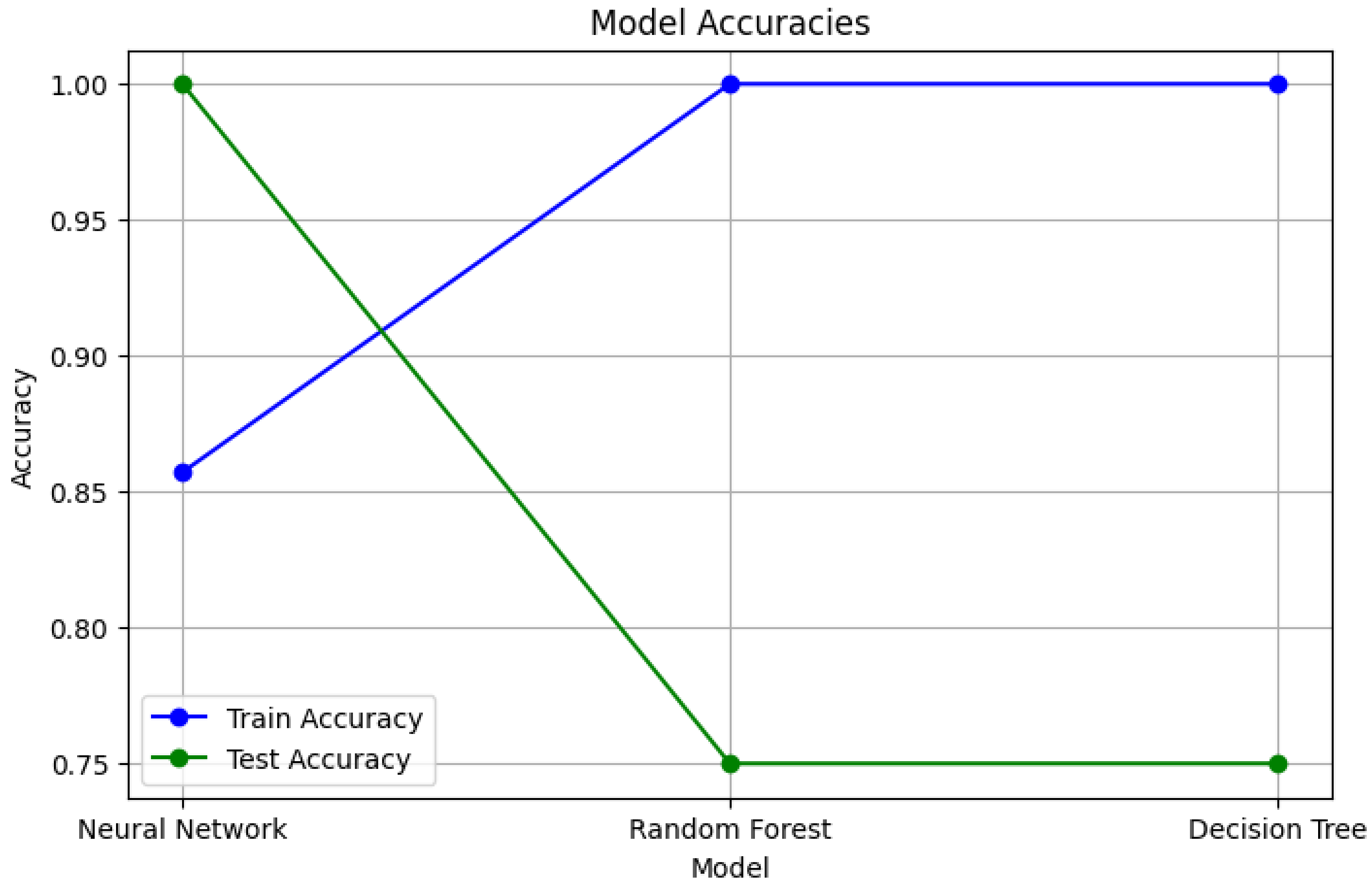
```
Decision Tree Training Accuracy: 1.0
Decision Tree Testing Accuracy: 0.75
```

In [42]:
```
## Print classification report
dt_report = classification_report(y_test, decision_tree_test_predictions)
print("Classification Report for Decision Tree:\n", dt_report)
```

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

        High       0.75      1.00      0.86         3
      Medium       0.00      0.00      0.00         1

    accuracy                           0.75         4
   macro avg       0.38      0.50      0.43         4
weighted avg       0.56      0.75      0.64         4
```

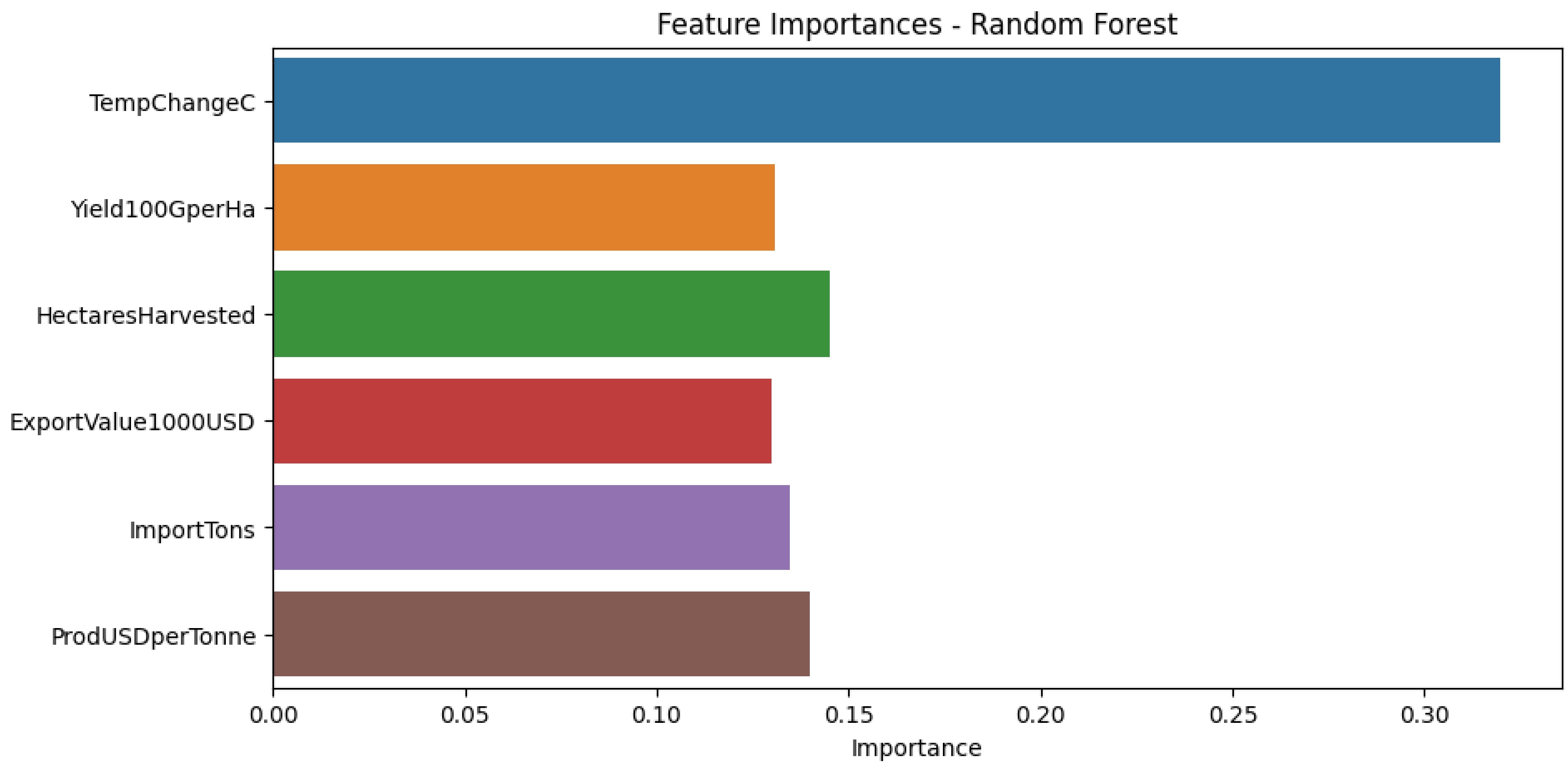
Plots of Accuracy Scores

In [43]:
```
## Plot all accuracy scores from models in one plot
models = ['Neural Network', 'Random Forest', 'Decision Tree']
train_accuracies = [neural_network_train_accuracy, random_forest_train_accuracy, decision_tree_train_accuracy]
test_accuracies = [neural_network_test_accuracy, random_forest_test_accuracy, decision_tree_test_accuracy]

# Plot train and test accuracies
plt.figure(figsize=(8, 5))
plt.plot(models, train_accuracies, marker='o', linestyle='-', label='Train Accuracy', color='b')
plt.plot(models, test_accuracies, marker='o', linestyle='-', label='Test Accuracy', color='g')
plt.title("Model Accuracies")
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



Feature Importance's for Random Forest

In [44]:
```
## Feature importances for the Random Forest model
feature_importances = random_forest.feature_importances_
feature_names = X.columns

plt.figure(figsize=(10, 5))
sns.barplot(y=feature_names, x=feature_importances)
plt.title("Feature Importances - Random Forest")
plt.xlabel("Importance")
plt.show()
```

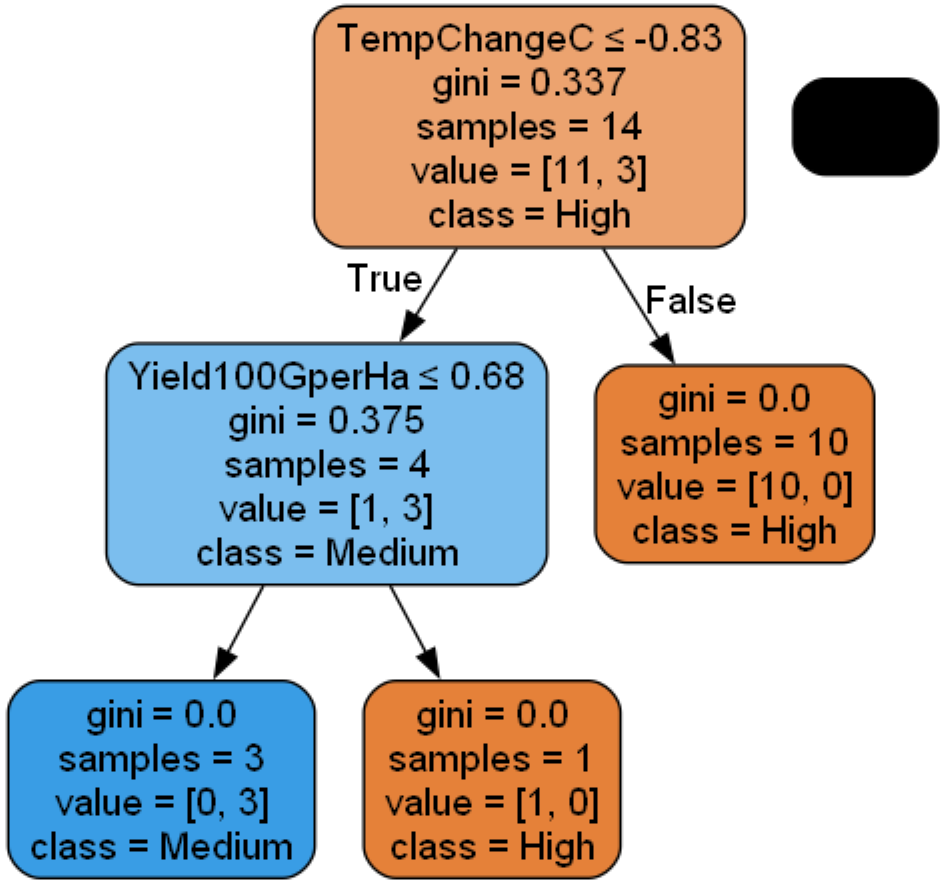


Visualizing the Decision Tree Model

In [45]:
```
## Export decision tree to dot format
dot_data = tree.export_graphviz(
    decision_tree,
    out_file=None,
    feature_names=X_train.columns,
    class_names=decision_tree.classes_.astype(str),
    filled=True,
    rounded=True,
    special_characters=True
)
```

```
## Convert dot data to a graph
graph = pydotplus.graph_from_dot_data(dot_data)

## Display decision tree as an image
Image(graph.create_png())
```

Out[45]:



In [46]:

```
# ## Create a decision tree visualization using dtreeviz
# viz = dtreeviz(
#     decision_tree,
#     X_train,
#     y_train,
#     target_name='WaterStressCategory',
#     feature_names=list(X_train.columns),
#     class_names=list(decision_tree.classes_.astype(str)),
```

```
#     title="Decision Tree - Water Stress Category"
# )

# ## Display decision tree visualization
# viz
```