Data Analysis on major power outage events in the continental U.S

By Saroop Samra, DSC Major UCSD, e: sksamra@ucsd.edu

Summary of Findings

Introduction

This dataset relates to major electricity outage patterns and characteristics for U.S. states from the time period Jan 2000-July 2016. The definition of a major outage is defined by the Department of Energy (DOE) as at least 50,000 customers or loss of at least 300 MW, and each observation in the dataset is a major outage event as described. The fundamental data variables include the reason of the outage (e.g. natural hazard), demand loss, customers affected and duration of the outage. This dataset was used in the paper, "A Multi-Hazard Approach to Assess Severe Weather-Induced Major Power Outage Risks in the U.S." (Mukherjee et al., 2018) [1]. The paper is based on a study that uses this dataset to develop a two-stage hybrid risk estimation model by using machine learning algorithms on this data. The model uses the various factors such as the type of natural hazards, rural versus urban data etc., to develop a model that could help state regulators to make risk informed resilience investment decisions.

The intent of this analysis is to predict the cause of a major power outage for natural disaster events. This model can be used to allow utility companies to determine the likely causes of an outage in the future given data from current observations such as the time of year as well as state based data climate characteristics, electrical consumption, economic indicators, land use characteristics of each state. This model then can help utility companies best prepare for outages and decide what is the best preventative measures they should take. For example, knowing that fire hazards have a high likelihood then utility companies can take preventative measures such as removing brushwood and tree trimming.

This prediction is a **classification** problem that given variables time of year as well as state-based data climate characteristics, electrical consumption, economic indicators will return the cause of a natural power outage (e.g. windstorms, fire etc.). We will take our data set and split in 70% for training and 30% for testing. The **objective** used is to the best accuracy (predicted for a sample must exactly match true values) as determined by the accuracy score for the testing data.

Before we start the generation of the model, we clean the data with the same techniques used in the earlier project. This for example includes cleanup of messy fields such as causes of natural disaster events for severe weather events. We also do some basic imputation for missingness of variables. Finally, we filter the data to just the outages that are related to natural disasters.

Baseline Model

The model developed uses 6 variables which we would know at the time of prediction, they are as follows:

- CLIMATE.CATEGORY : This represents the climate episodes corresponding to the years. The categories — “Warm”, “Cold” or “Normal” episodes of the climate are based on a threshold of 7 0.5 °C for the Oceanic Niño Index (ONI). This is a nominal variable where we use one-shot encoding. The rationale for using the climate category is that certain climates are have higher frequency of specific types of natural disasters, such as the hurricanes in Florida.

- ANOMALY.LEVEL : This represents the oceanic El Niño/La Niña (ONI) index. This is a quantitative variable where we use a z-scale transformation. The rationale for using this is this represents issues relating to the climate patterns.
- CLIMATE.REGION : U.S. Climate regions as specified by National Centers for Environmental Information (nine climatically consistent regions in continental U.S.A.). This is a nominal variable where we use one-shot encoding. The rationale for using the climate region is that certain climates are have higher frequency of specific types of natural disasters, such as the hurricanes in Florida.
- POPDEN_RURAL : Population density of the rural areas (persons per square mile). This is a quantitative variable where we use a z-scale transformation. The rationale for using this is the natural disasters might be of different natures in rural populations as opposed to urban areas. For example, wildfire might not be an issue in a state dominated by urban cities and few rural areas.
- PC.REALGSP.USA : Per capita real GSP in the U.S. (measured in 2009 chained U.S. dollars). This is a quantitative variable where we use a z-scale transformation. The rationale for using this is the natural disasters might be affected by different regional economic factors. For example, states with low GSP might not be able to invest in improvements to the utility infrastructure which would exacerbate large power outages.
- TOTAL.CUSTOMERS : Annual number of total customers served in the U.S. state. This is a quantitative variable where we use a z-scale transformation. The rationale for using this is the natural disasters might be affected by different electric consumption patterns. For example, states with lots of customers may be in certain areas of the USA, such as coastal regions, and be more prone to specific natural disaster causes. Furthermore, for large install base of customers, the utility company made be able to afford to make investments in dealing with natural disaster planning and thus the event may not have caused a major impact and be classified as a major power outage.

Note, we tried to use the U.S. State as a variable; however, there was insufficient data for states which caused issues when the test data had a state that was not in the training data. For that reason, we excluded the U.S. State variable from our model to make it more robust.

As mentioned, these are all variables that can be known at the time of prediction. We explicitly did not want to use variables such as the duration of the outage, the loss in power or the customers affected as these are not available at the time of prediction.

The balance for this model was to not have too many variables and to keep it light to have good performance. I used a Decision Tree selected a maximum depth of 20. The resulting model had a training data accuracy of 95.2% and a test data accuracy of 49.7%. Furthermore, we tested the time performance of the prediction and found it could predict over 1000 events in 15.6 ms.

As we expected the training test accuracy was much higher than the test data accuracy. Overall the performance of calculations was fast and even though the accuracy with the training data was excellent, the accuracy with test data was poor, only 49.7% of the time did it produce correct results, which is no better than just a random guess.

Final Model

The model developed extended the base model with an additional 18 variables, care was taken to make sure all these variables were known at the time of prediction. The additional variables are as follows:

- MONTH : Indicates the month when the outage event occurred. This is an ordinal encoding, the rationale for using the month is that certain natural disasters are dependent on the time of year, such as snow storms in winter or fires in summer.

The following variables were related to state electricity consumption information:

- RES.CUSTOMERS : Annual number of customers served in the residential electricity sector of the U.S. state
- COM.CUSTOMERS : Annual number of customers served in the commercial electricity sector of the U.S. state
- IND.CUSTOMERS : Annual number of customers served in the industrial electricity sector of the U.S. state
- IND.SALES : Electricity consumption in the industrial sector (megawatt-hour)
- RES.PRICE : Monthly electricity price in the residential sector (cents/kilowatt-hour)
- COM.PERCEN : Percentage of commercial electricity consumption compared to the total electricity consumption in the state (in %)

The state electricity consumption data were all quantitative variable where we use a z-scale transformation. The rationale for using consumption variables is high amounts of usage of electricity could be a precursor to natural disasters, for example high electrical usage might indicate cold weather or extreme hot weather.

The following variables were related to regional land characteristics:

- PCT_LAND : The Percentage of land area in the U.S. state as compared to the overall land area in the continental U.S. This is a quantitative variable where we use a z-scale transformation. The rationale for using the land percentage is that larger states have potentially different types of natural disaster events as opposed to smaller land states.
- AREAPCT_URBAN : Percentage of the land area of the U.S. state represented by the land area of the urban clusters (in %)
- POPDEN_UC : Population density of the urban clusters (persons per square mile)
- AREAPCT_URBAN : Percentage of the land area of the U.S. state represented by the land area of the urban areas (in %)
- PCT_WATER_INLAND : Percentage of inland water area in the U.S. state as compared to the overall inland water area in the continental U.S. (in %)
- POPPCT_URBAN : Percentage of the total population of the U.S. state represented by the urban population (in %)
- POPPCT_UC : Percentage of the total population of the U.S. state represented by the population of the urban clusters (in %)

The regional land characteristics data were all quantitative variable where we use a z-scale transformation. The rationale for using land variables is this represents issues relating to the usage of land that could impact natural disasters, for example areas with high water or urban vs rural land usage could have different frequencies of certain types of natural disasters.

The following variables were related to regional economic characteristics:

- PC.REALGSP.REL : Relative per capita real GSP as compared to the total per capita real GDP of the U.S. (expressed as fraction of per capita State real GDP & per capita US real GDP)
- UTIL.CONTRI : Utility industry's contribution to the total GSP in the State (expressed as percent of the total real GDP that is contributed by the Utility industry) (in %)
- PC.REALGSP.CHANGE : Percentage change of per capita real GSP from the previous year (in %)

The regional economic characteristics data were all quantitative variable where we use a z-scale transformation. The rationale for using these economic variables is the strength of the state economy could impact how much investment utilities could make to improve infrastructure which may reduce the cases when a natural disaster would result in a large power outage.

As mentioned, these are all variables that can be known at the time of prediction.

The balance for this model was it had a significantly higher number of variables (4 times more) and we would favor accuracy over computation performance. I did an expanded analysis of an additional two classifications: including K-Neighbors and Random Forest. For each K-Neighbors classification the arguments were tested to find the optimal results, number of neighbors from 5 to 80. For Random Forest classification the arguments were tested to find the optimal results was maximum depth between 60-90 and the number of features tested between 40-80, this was a time-consuming process as the search space was O(N^2) and it took over an hour. Finally, for the original Decision Tree classification the arguments were tested to find the optimal results was maximum depth between 50-100.

The code is shown below to iterate the search space for all 3 classification schemes. The results showed that Random Forest classification was better the Decision Tree classification and the worst was K-Neighbors classification. The best solution for Random Forest classification scheme was the number of estimators set to 65 and the maximum depth set to 74. The resulting model had a training data accuracy of 96.4.% and a test data accuracy of 74.9%. Furthermore, we tested the time performance of the prediction and found it could predict over 1000 events in 77.6 ms.

As we expected the training test data accuracy was much higher than the test data accuracy. Overall the performance of calculations was 5x slower than the base model and the training accuracy was similar, however, critically the accuracy on the test data had improved by 51%. The resulting testing accuracy was 74.9%. In conclusion even with overall increase in computation time, the final model was much more robust as it had higher accuracy, three quarters of the time it would predict the correct result.

Fairness Evaluation

The fairness of the model is evaluated in two different evaluation experiments.

Permutation Fairness Evaluation

We take two climate characteristic variables (ANOMALY.LEVEL and CLIMATE.REGION) and do a permutation of these in the training data, that is we shuffle them but leave all other variables including the classified column the same. We do this for each simulation 1000 times, and our test statistic is finding the difference in the accuracy score compared with the original accuracy of the (non-permutated) original test data using the same model.

- The Null Hypothesis is as follows: "The model accuracy does not improve when we have shuffled the climate characteristic variables (ANOMALY.LEVEL and CLIMATE.REGION)."
- The Alternative Hypothesis is as follows: "The model accuracy improves with climate characteristic variables (ANOMALY.LEVEL and CLIMATE.REGION)."
- The test significance is 5 percent

The code and the simulation results are below and show a p-value of 0.022. Figure 1 shows the histogram of the experiment. The p-value is less than test significance which means we can reject the Null Hypothesis and the model provides fairer results when we include the climate characteristic variables (ANOMALY.LEVEL and CLIMATE.REGION).

Fairness: Evaluating for Bias

As well as doing a permutation test, we also performed analysis to see if the model had any biases. For each fairness experiment we plotted a bar chart of the accuracy across a range of values for a specific variable. This allows us to check if the model is more or less accurate for a subset of the values for that variable.

- Figure 2 : Climate Category (CLIMATE.CATEGORY) Fairness Accuracy. We first look at the climate category (CLIMATE.CATEGORY) and plot the accuracy of the cold, normal and warm climates. Normal climates have the highest accuracy, of approximately 80%, but warm climates are less accurate, less than 70%. However, this might just indicate that the data set has a lot of outages in the colder climates.
- Figure 3 : Month (MONTH) Fairness Accuracy. The next fairness we look at is for the MONTH variable. The plot shows the model is most accurate for the month of January receiving a 95% accuracy, yet in March the accuracy is less than 0%! Again we should confirm this is not just because the data set has less outages in March.

Rather than evaluating fairness for ordinal and nominal data sets as above, we can do the same for quantitative variables by placing them into buckets using the panda qcut function. We bucket into three for the following quantitative columns so we can evaluate if the model is more accurate for the lower one-third range, the upper one-third or the middle one-third range of the data.

- Figure 4 : Urban Land Area Clusters (AREAPCT_URBAN) Fairness Accuracy. The Urban Land Area Clusters (AREAPCT_URBAN) Fairness Accuracy show that the model is most accurate for top one-third range. It achieves approximately 80% accuracy yet for the bottom one-third range it is approximately 50% accuracy.
- Figure 5: Real GSP Rel (PC.REALGSP.REL) Fairness Accuracy. The financial metric of relative Real Gross State Product (PC.REALGSP.REL) fairness evaluation chart shows that the states with the lowest relative GSP are more accurate to predict. The top one-third GSP based states have only an accuracy of less than 60%.
- Figure 6 : Rural Population Density (POPDEN_RURAL) Fairness Accuracy. The plot shows the top one-third of rural density states have a model accuracy close to 80%. But the model has poor accuracy for states which have low rural population densities and is only 50% accurate.
- Figure 7 : Model Correctness by Cause. The distribution of the correctness over the causes shows that the model does well for storms and winter, where there is most of all the data but does poory for earthquake causes. For example, 60 storm events are corrected correctly yet less than 5 are incorrect yet for earthquake it got 0% correct albiet there was only 1 earthquake cause. This is not an unsuprising result because the model will do better where there is a lot of test data, and for example storms was a frequency outage cause yet there were very few earthquake causes.
- Figure 8: Model Correctness of Duration vs RES.PRICE. Thr correctness is show mapped on also showing that short duration outages occur when residential price is Low. Long durations outliers when Residential Price is Higher. The model has better accuracy when the duration is longer.
- Figure 9: Model Correctness PCT_LAND vs POPULATION. - The scatter plot shows a positive trend, and the incorrect predictions are clustered around populations under 1 million.
- Figure 10: Model Correctness POPPCT_URBAN vs POPULATION. The scatter plot shows a positive trend, and the incorrect predictions have a cluster around 70% urban density with state populations under 1 million.
- Figure 11: Model Correctness Duration of Major Outages over time. We see that over the time period, 2000-2015, the model will predict more cases correct each year than. The model does not appear to bias around the year.

- Figure 12: Model Correctness Customers Affected of Major Outages over time. We see that over the time period, 2000-2015, the model predicts more cases correct each year. In 2014 the model predicted correctly for cases that affected 90% of the customers that year. In 2012, there were less customers affected but the model was almost 100% correct. The year 2008 the model accuracy is better than 50%, however, it does show a large number of customers who were affected and the model did not determine the right cause.

Further Work

The fairness study has shown a range of variables have better or worse accuracy depending on their actual values. A follow up study could be done to establish the cause of why the model behaves poorly for some values of these variables and ultimately establish how to improve the model to avoid this bias.

Citations

[1] A Multi-Hazard Approach to Assess Severe Weather-Induced Major Power Outage Risks in the U.S." (Mukherjee et al., 2018) https://www.sciencedirect.com/science/article/pii/S0951832017307767 (https://www.sciencedirect.com/science/article/pii/S0951832017307767)

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [2]:

```
# Clean Data: load the Outages, and skip first few rows
outages_fp = os.path.join("outages_data", "outage.xlsx")
outages = pd.read_excel(outages_fp, skiprows=5)
# Drop the sub header row
outages = outages.drop(0)
# Drop the empty variables column
outages = outages.drop("variables", axis=1)
outages.head()
```

Out[2]:

	OBS	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMA
1	1.0	2011.0	7.0	Minnesota	MN	MRO	East North Central	
2	2.0	2014.0	5.0	Minnesota	MN	MRO	East North Central	
3	3.0	2010.0	10.0	Minnesota	MN	MRO	East North Central	
4	4.0	2012.0	6.0	Minnesota	MN	MRO	East North Central	
5	5.0	2015.0	7.0	Minnesota	MN	MRO	East North Central	

5 rows × 56 columns

In [3]:

```
# Clean Data: TODO : Uncomment Drop OBS Column
outages = outages.drop("OBS", axis=1)

# Clean Data: Drop these: 9 Missing Months, Outages Start, Duration & Restoration, ANOMAL
Y.LEVEL, CLIMATE.CATEGORY, RES.PRICE-IND.PERCEN
outages = outages[outages["MONTH"].notnull()]

# Clean Data: Cleanup messy field to aggerate types of severe weather events
outages = outages.replace({"thunderstorm; islanding" : "thunderstorm",
                           "wind storm" : "wind",
                           "wind/rain" : "wind",
                           "heavy wind" : "wind",
                           "winter storm" : "winter",
                           "snow/ice storm" : "winter",
                           "snow/ice " : "winter",
                           "snow" : "winter",
                           "hailstorm" : "winter",
                           "uncontrolled loss" : "other",
                           "public appeal" : "other",
                           "fog" : "other",
                           "heatwave" : "fire",
                           "wildfire" : "fire",
                           "hurricane" : "hurricane/tornadoes",
                           "hurricanes" : "hurricane/tornadoes",
                           "tornadoes" : "hurricane/tornadoes",
                           "thunderstorm" : "storm",
                })

# Clean Data: Cleanup messy field remove spaces at start
outages["CAUSE.CATEGORY.DETAIL"] = outages["CAUSE.CATEGORY.DETAIL"].str.strip(" ")
```

In [4]:

```
# Limit analysis to Natural Disasters (severe weather events)
outages = outages[outages["CAUSE.CATEGORY"]  == "severe weather"]
```

In [5]:

```
# Fill Missing Data
outages = outages.fillna(outages.mean())
# Drop data drom non recorded cause or regions
outages = outages.dropna(subset=["CAUSE.CATEGORY.DETAIL", "CLIMATE.REGION"])
outages = outages.reset_index(drop=True)
```

Baseline Model

In [6]:

```
# z-scale for quantative data
def z_scale(x):
    return (x-x.mean())/x.std()
# column transformer
column_transformer = ColumnTransformer(
    transformers = [
    ('Climate Category', OneHotEncoder(categories='auto'), ["CLIMATE.CATEGORY"]),
    ('Anomaly Level', FunctionTransformer(func=z_scale, validate=True), ["ANOMALY.LEVEL"
]),
    ('Climate Region', OneHotEncoder(categories='auto', handle_unknown='ignore'), ["CLIMAT
E.REGION"]),
    ('Rural Population Density', FunctionTransformer(z_scale, validate=True), ["POPDEN_RUR
AL"]),
    ('Per Capita GDP US', FunctionTransformer(func=z_scale, validate=True), ["PC.REALGSP.U
SA"]),
    ('Total Customers', FunctionTransformer(func=z_scale, validate=True), ["TOTAL.CUSTOMER
S"]),
    ],
    remainder = 'drop')
# create pipeline
pipeline = Pipeline([
    ('preprocessing', column_transformer),
    ('classifier', DecisionTreeClassifier(max_depth=20))
])
```

In [7]:

```
# target
target = "CAUSE.CATEGORY.DETAIL"
# Split Testing Data
test_size=0.3
x_train, x_test, y_train, y_test = train_test_split(outages.drop(target, axis=1), outages[
target], test_size=test_size)
```

In [79]:

```
# Train
pipeline.fit(x_train, y_train)
# Accuracy
train_score = accuracy_score(y_train, pipeline.predict(x_train), normalize=True, sample_we
ight=None)
test_score = accuracy_score(y_test, pipeline.predict(x_test), normalize=True, sample_weigh
t=None)
print("Training Data Accuracy =", train_score, " Test Data Accuracy =", test_score)
%timeit pipeline.predict(x_train)
```

```
Training Data Accuracy = 0.9522613065326633  Test Data Accuracy = 0.497076023
39181284
15.6 ms ± 1.27 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Final Model

In [8]:

```
column_transformer = ColumnTransformer( transformers = [
    ('Climate Region', OneHotEncoder(categories='auto', handle_unknown='ignore'), ["CLIMAT
E.REGION"]),
    ('Climate Category', OneHotEncoder(categories='auto'), ["CLIMATE.CATEGORY"]),
    ('Anomaly Level', FunctionTransformer(func=z_scale, validate=True), ["ANOMALY.LEVEL"
]),
    ('Rural Population Density', FunctionTransformer(func=z_scale, validate=True), ["POPDE
N_RURAL"]),
    ('Per Capita GDP US', FunctionTransformer(func=z_scale, validate=True), ["PC.REALGSP.U
SA"]),
    ('Total Customers', FunctionTransformer(func=z_scale, validate=True), ["TOTAL.CUSTOMER
S"]),
    ('Month', OrdinalEncoder(categories=[[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.
0, 11.0, 12.0]]), ["MONTH"]),
    ('Residential Price', FunctionTransformer(func=z_scale, validate=True), ["RES.PRICE"
]),
    ('Commercial Consumption', FunctionTransformer(func=z_scale, validate=True), ["COM.PER
CEN"]),
    ('Industrial Customers', FunctionTransformer(func=z_scale, validate=True), ["IND.CUSTO
MERS"]),
    ('Total Electricity Consumption', FunctionTransformer(func=z_scale, validate=True), [
"TOTAL.SALES"]),
    ('Commercial Customers', FunctionTransformer(func=z_scale, validate=True), ["COM.CUSTO
MERS"]),
    ('Residential Customers', FunctionTransformer(func=z_scale, validate=True), ["RES.CUST
OMERS"]),
    ('Land %', FunctionTransformer(func=z_scale, validate=True), ["PCT_LAND"]),
    ('Urban Land %', FunctionTransformer(func=z_scale, validate=True), ["AREAPCT_URBAN"]),
    ('Urban Population Area Clusters', FunctionTransformer(func=z_scale, validate=True), [
"POPDEN_UC"]),
    ('Urban Land Area Clusters', FunctionTransformer(func=z_scale, validate=True), ["AREAP
CT_URBAN"]),
    ('Industrial Consumption', FunctionTransformer(func=z_scale, validate=True), ["IND.SAL
ES"]),
    ('Inland Water Area Customers', FunctionTransformer(func=z_scale, validate=True), ["PC
T_WATER_INLAND"]),
    ('Urban Population Density', FunctionTransformer(func=z_scale, validate=True), ["POPPC
T_URBAN"]),
    ('Urban Population', FunctionTransformer(func=z_scale, validate=True), ["POPPCT_UC"]),
    ('Utility Contribution', FunctionTransformer(func=z_scale, validate=True), ["UTIL.CONT
RI"]),
    ('Real GSP Rel', FunctionTransformer(func=z_scale, validate=True), ["PC.REALGSP.REL"
]),
    ('Real GSP Change', FunctionTransformer(func=z_scale, validate=True), ["PC.REALGSP.CHA
NGE"]),
    ], remainder = 'drop')
```

In [10]:

```
# create pipeline
pipeline = Pipeline([
    ('preprocessing', column_transformer),
    ('classifier', RandomForestClassifier(max_depth=74, n_estimators=65))
])
```

In [120]:

```
# Train
pipeline.fit(x_train, y_train)

# Accuracy
train_score = accuracy_score(y_train, pipeline.predict(x_train), normalize=True, sample_we
ight=None)
test_score = accuracy_score(y_test, pipeline.predict(x_test), normalize=True, sample_weigh
t=None)

print("Training Data Accuracy =", train_score, " Test Data Accuracy =", test_score)

%timeit pipeline.predict(x_train)
```

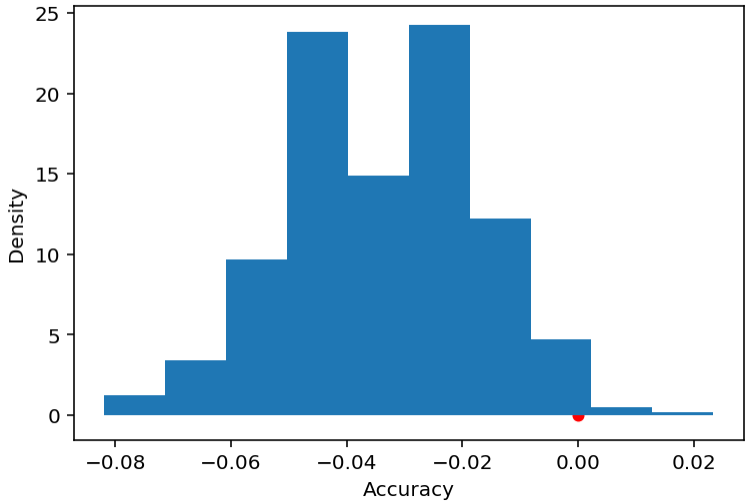
```
Training Data Accuracy = 0.964824120603015  Test Data Accuracy = 0.7485380116
959064
77.6 ms ± 5.01 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Fairness Evaluation

In [11]:

```
def permutation_statistic(px_test, py_test, original_score):
    # Accuracy
    sample_score = accuracy_score(py_test, pipeline.predict(px_test), normalize=True, samp
le_weight=None)
    return sample_score - original_score

def shuffle(df):
    # shuffle climate characteristics data
    climate = ["CLIMATE.CATEGORY", "ANOMALY.LEVEL"]
    for column in climate:
        df[column] = df[column].sample(frac=1).to_list()
    return df

observed_stat = permutation_statistic(x_test, y_test, test_score)
sample_means = []
num_samples = 1000
for _ in range(num_samples):
    px_test = shuffle(x_test.copy())
    sample_stat = permutation_statistic(px_test, y_test, test_score)
    sample_means.append(sample_stat)

p_value = np.count_nonzero(sample_means>=observed_stat)/num_samples

fignum = 1
pd.Series(sample_means).plot(kind='hist', title="Fig. "+str(fignum)+': Permutation Test of
climate variables pvalue=' + str(p_value), density=True)
plt.xlabel('Accuracy')
plt.ylabel('Density')
plt.scatter([observed_stat], [0], c='r', s=25);
fignum += 1
```

Fig. 1: Permutation Test of climate variables pvalue=0.022



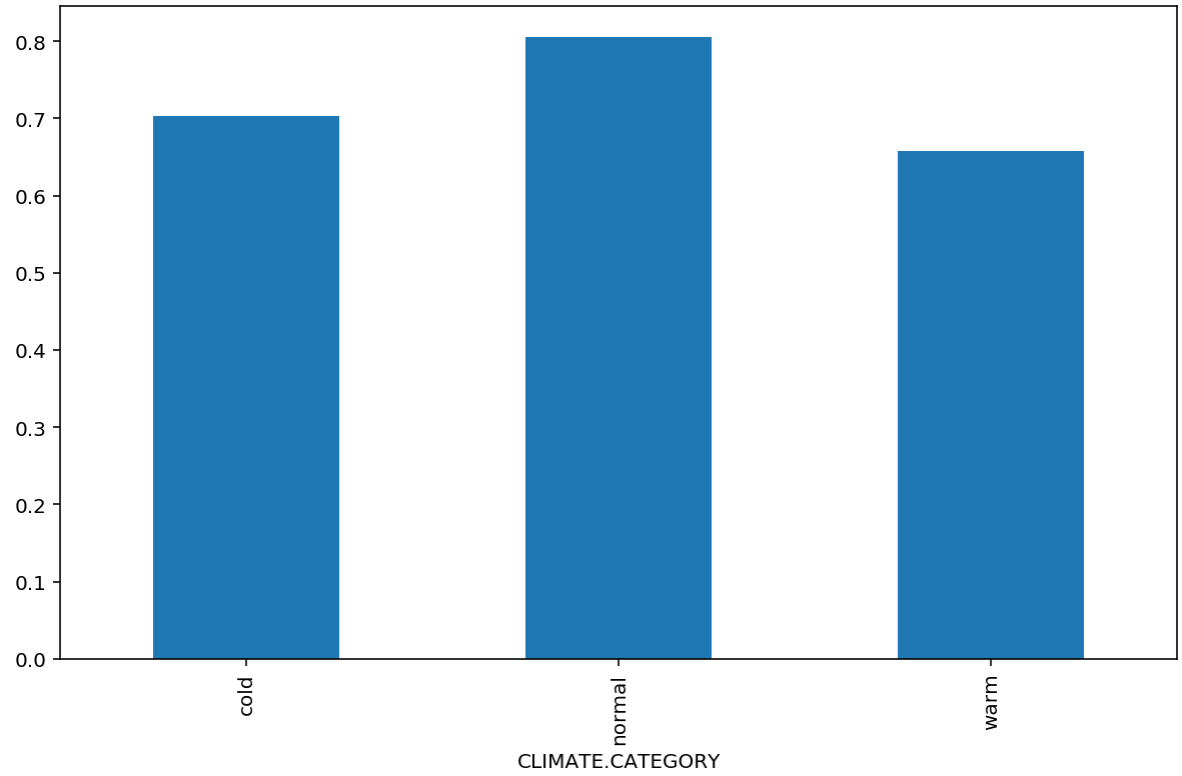
In [12]:

```
# Test accuracy
combined_test_data = x_test.copy()
combined_test_data[target] = y_test
def eval_model_accuracy(df):
    px_test = df.copy().drop(target, axis=1)
    py_test = df[target]
    accuracy = accuracy_score(py_test, pipeline.predict(px_test), normalize=True, sample_w
eight=None)
    return accuracy
```

In [13]:

```
chart = combined_test_data.groupby("CLIMATE.CATEGORY").apply(eval_model_accuracy).plot(kin
d='bar', figsize=(10,6), title="Fig. "+str(fignum)+': Climate Category (CLIMATE.CATEGORY)
 Fairness Accuracy')
fignum += 1
```

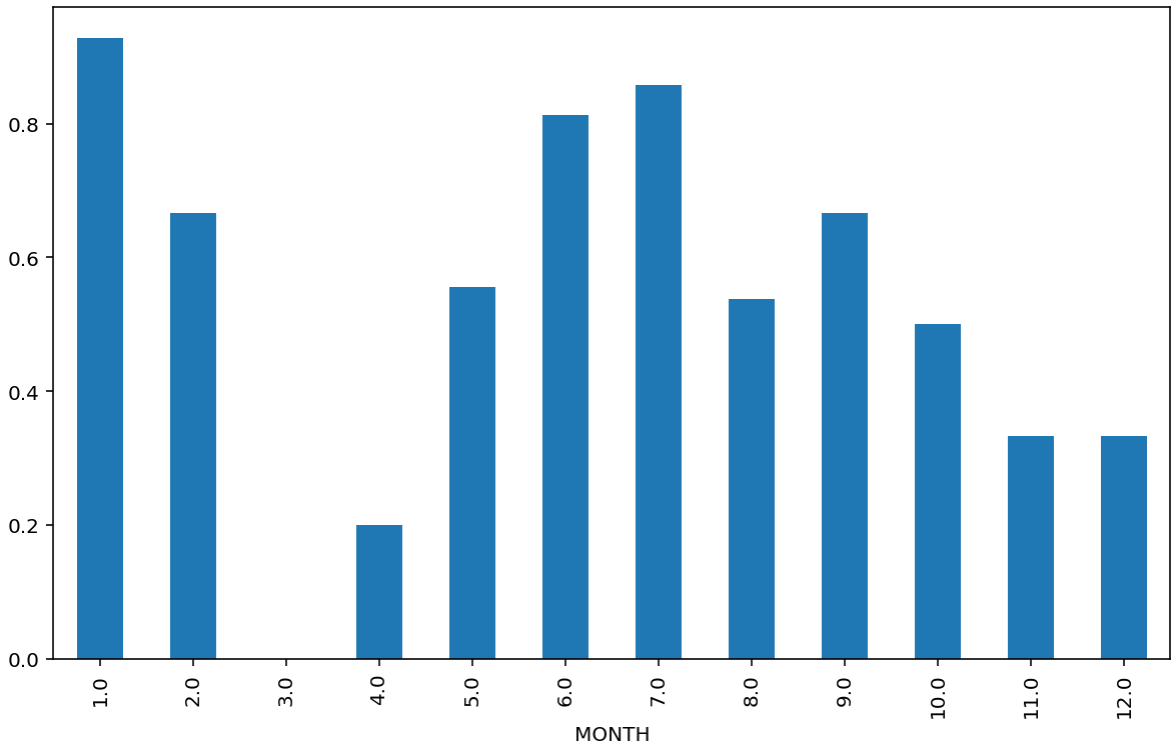
Fig. 2: Climate Category (CLIMATE.CATEGORY) Fairness Accuracy



In [14]:

```
chart = combined_test_data.groupby("MONTH").apply(eval_model_accuracy).plot(kind='bar', fi
gsize=(10,6), title="Fig. "+str(fignum)+': Month (MONTH) Fairness Accuracy')
fignum += 1
```

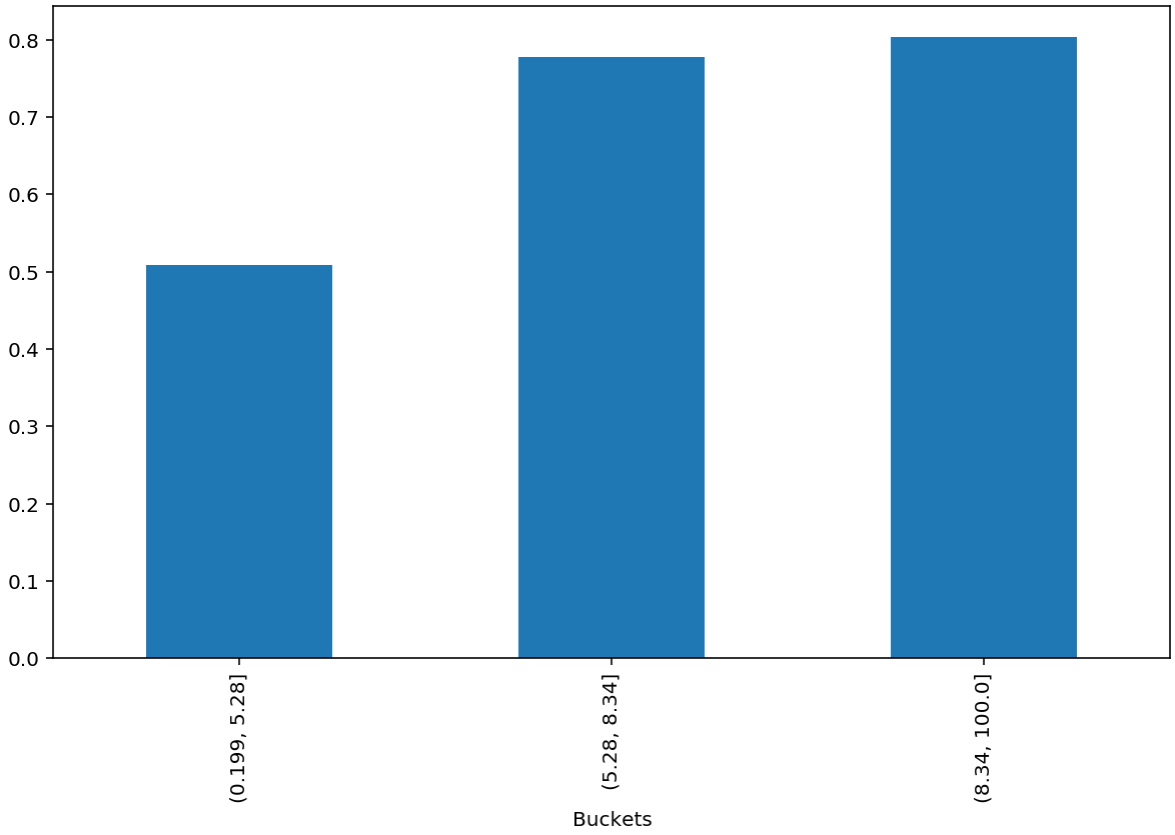
Fig. 3: Month (MONTH) Fairness Accuracy



In [15]:

```
df2 = combined_test_data.copy()
df2["Buckets"] = pd.qcut(combined_test_data["AREAPCT_URBAN"], 3)
chart = df2.groupby("Buckets").apply(eval_model_accuracy).plot(kind='bar', figsize=(10,6),
title="Fig. "+str(fignum)+': Urban Land Area Clusters (AREAPCT_URBAN) Fairness Accuracy')
fignum += 1
```

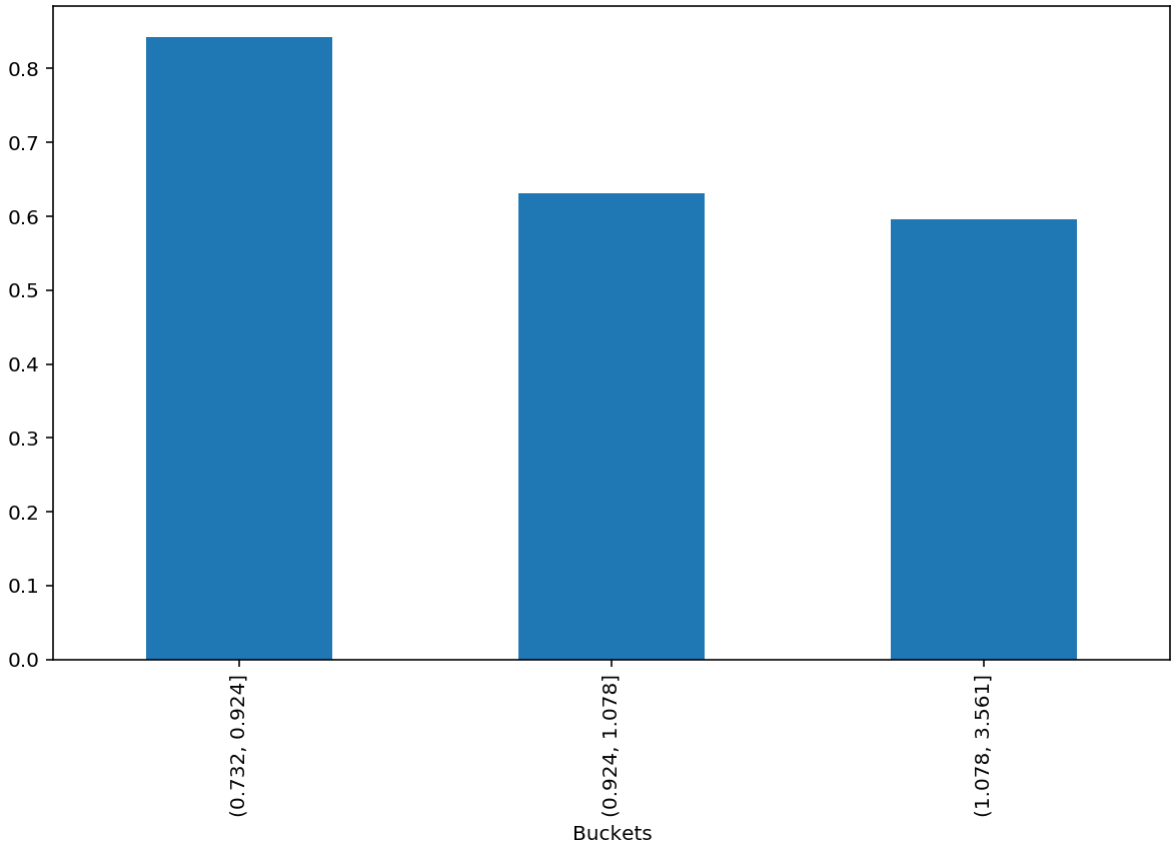
Fig. 4: Urban Land Area Clusters (AREAPCT_URBAN) Fairness Accuracy



In [16]:

```
df2["Buckets"] = pd.qcut(combined_test_data["PC.REALGSP.REL"], 3)
chart = df2.groupby("Buckets").apply(eval_model_accuracy).plot(kind='bar', figsize=(10,6),
title="Fig. "+str(fignum)+': Real GSP Rel (PC.REALGSP.REL) Fairness Accuracy')
fignum += 1
```

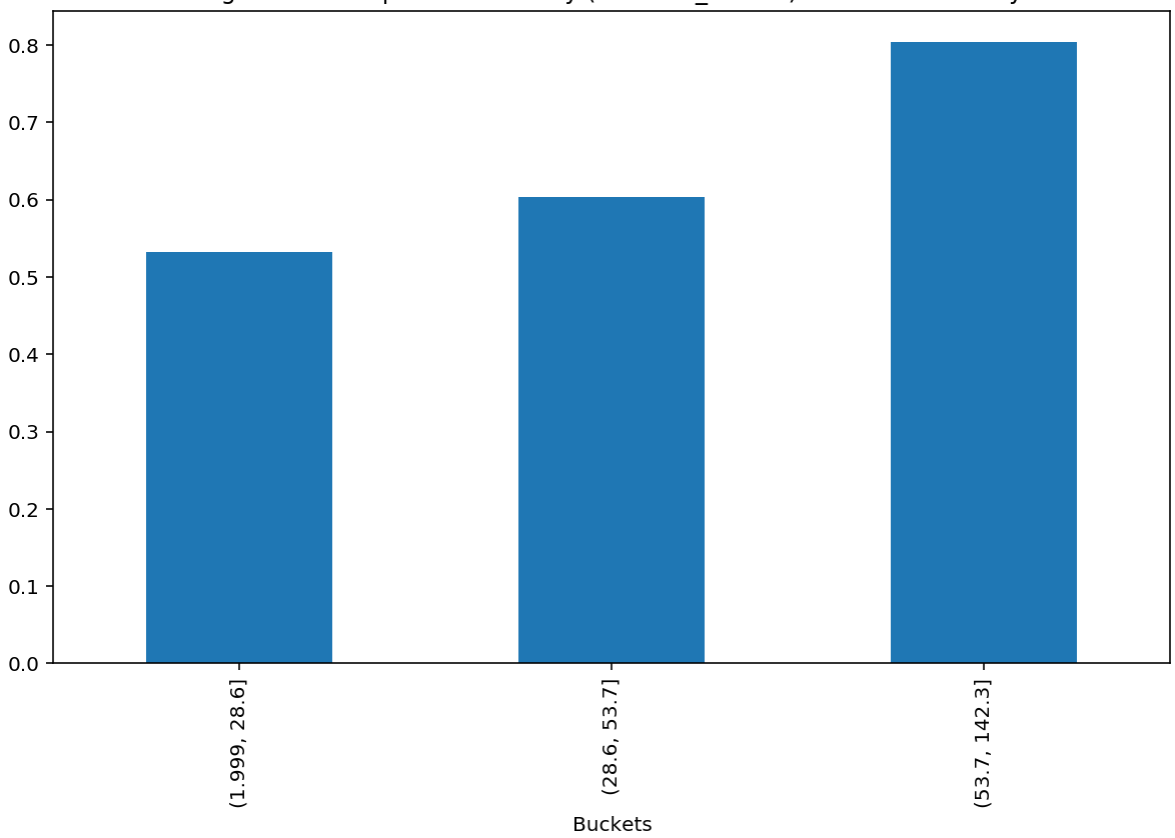
Fig. 5: Real GSP Rel (PC.REALGSP.REL) Fairness Accuracy



In [17]:

```
df2["Buckets"] = pd.qcut(combined_test_data["POPDEN_RURAL"], 3)
chart = df2.groupby("Buckets").apply(eval_model_accuracy).plot(kind='bar', figsize=(10,6),
title="Fig. "+str(fignum)+': Rural Population Density (POPDEN_RURAL) Fairness Accuracy')
fignum += 1
```

Fig. 6: Rural Population Density (POPDEN_RURAL) Fairness Accuracy

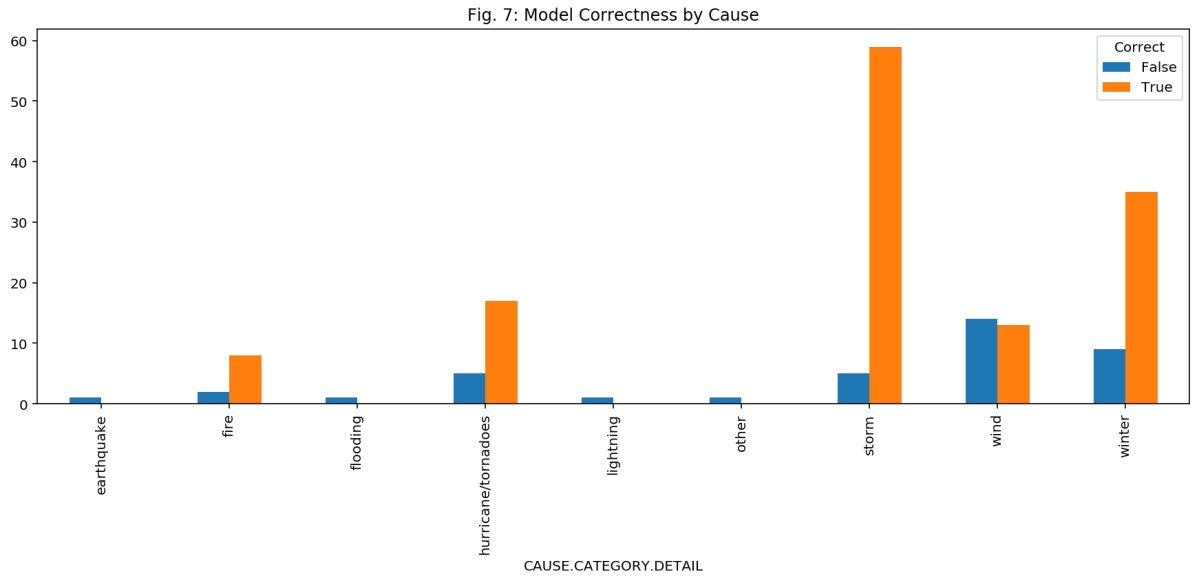


In [ ]:

```
# Add Column for correctness
y_pred = pipeline.predict(x_test)
combined_test_data["Correct"] = y_pred == y_test
```

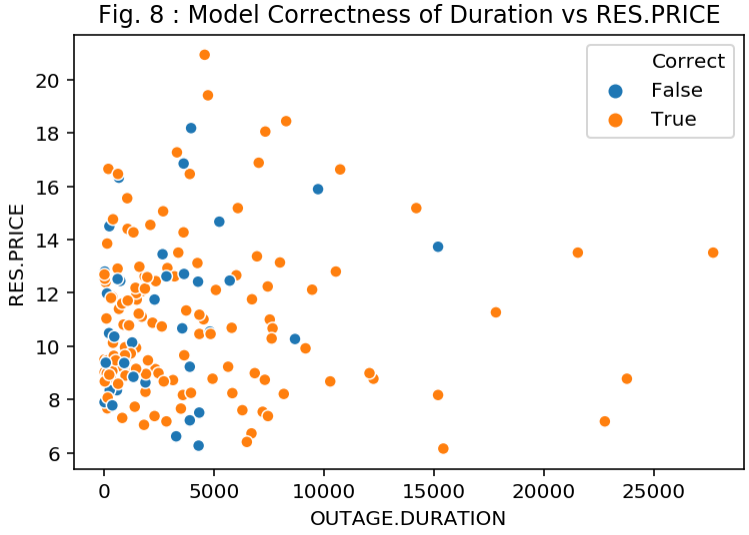
In [60]:

```
df = combined_test_data.pivot_table(index=target, columns="Correct", values="OUTAGE.DURATION", aggfunc="count", fill_value=0)
df.plot(kind="bar", figsize=(15,5))
title = plt.title("Fig. "+str(fignum)+": "+"Model Correctness by Cause", loc='center', pad=None)
fignum += 1
```



In [66]:

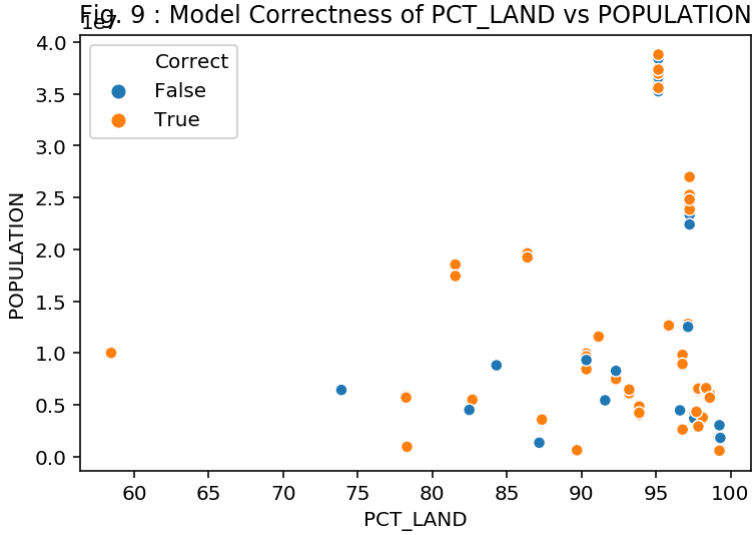
```
sns.scatterplot(data=combined_test_data,x='OUTAGE.DURATION', y='RES.PRICE', hue="Correct")
title = plt.title("Fig. "+str(fignum)+ " : Model Correctness of Duration vs RES.PRICE")
fignum+=1
```



In [62]:

```
sns.scatterplot(data=combined_test_data,x='PCT_LAND', y='POPULATION', hue="Correct")
title = plt.title("Fig. "+str(fignum)+ " : Model Correctness of PCT_LAND vs POPULATION")
fignum+=1
```

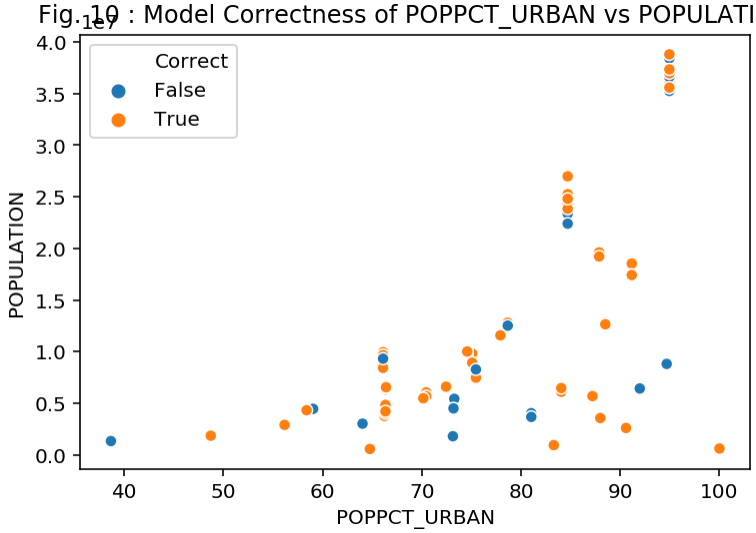
Fig. 9 : Model Correctness of PCT_LAND vs POPULATION



In [63]:

```
sns.scatterplot(data=combined_test_data, x='POPPCT_URBAN', y='POPULATION', hue="Correct")
title = plt.title("Fig. "+str(fignum)+ " : Model Correctness of POPPCT_URBAN vs POPULATIO
N")
fignum+=1
```

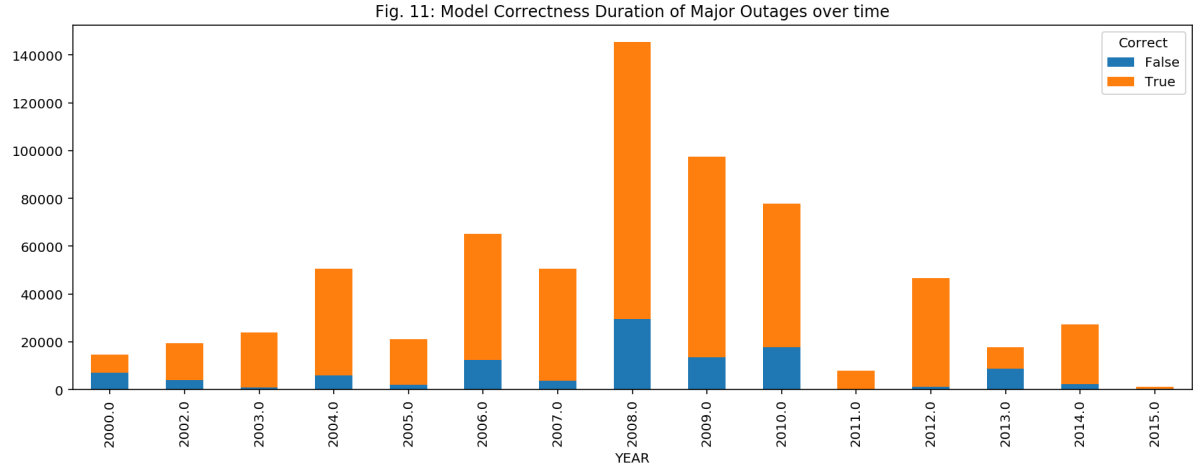
Fig. 10 : Model Correctness of POPPCT_URBAN vs POPULATION



In [64]:

```
df = combined_test_data.pivot_table(index="YEAR", columns="Correct", values="OUTAGE.DURATION", aggfunc="sum", fill_value=0)
df.plot(kind="bar", figsize=(15,5), stacked=True)
title = plt.title("Fig. "+str(fignum)+": "+"Model Correctness Duration of Major Outages over time", loc='center', pad=None)
fignum += 1
```

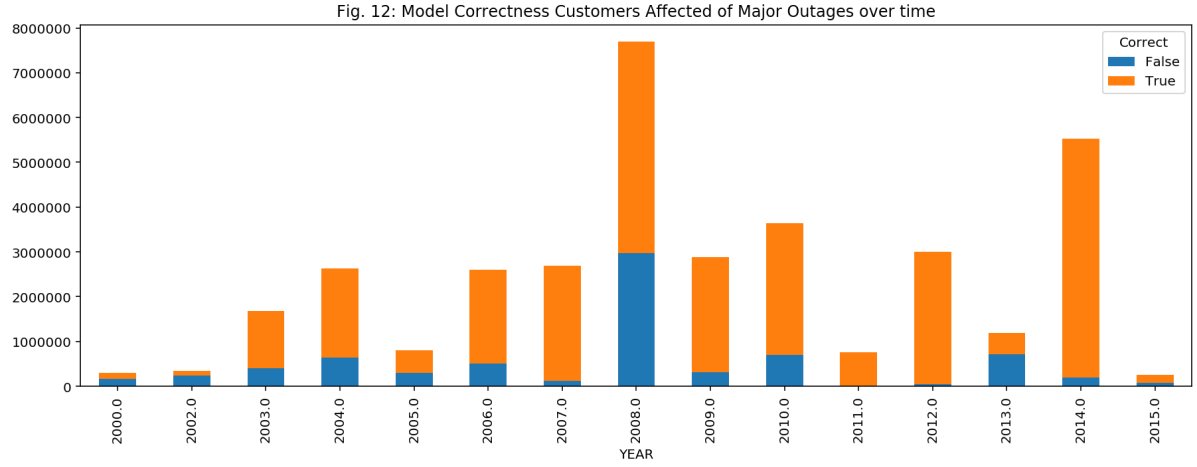
Fig. 11: Model Correctness Duration of Major Outages over time



In [65]:

```
df = combined_test_data.pivot_table(index="YEAR", columns="Correct", values="CUSTOMERS.AFFECTED", aggfunc="sum", fill_value=0)
df.plot(kind="bar", figsize=(15,5), stacked=True)
title = plt.title("Fig. "+str(fignum)+": "+"Model Correctness Customers Affected of Major Outages over time", loc='center', pad=None)
fignum += 1
```

Fig. 12: Model Correctness Customers Affected of Major Outages over time



Code Reference

Random Forest Classifier Searching Best coefficients

In [101]:

```
# Find best coefficients for RandomForestClassifier
x_train, x_test, y_train, y_test = train_test_split(outages.drop(target, axis=1), outages[
target], test_size=test_size)
result = pd.DataFrame({"max_depth" : [], "n_estimators" : [], "train_score" : [], "test_sc
ore" : []})
for max_depth in range(60, 91):
    for n_estimators in range(40,81):
        pipeline = Pipeline([
            ('preprocessing', column_transformer),
            ('classifier', RandomForestClassifier(max_depth=max_depth, n_estimators=n_esti
mators))
        ])
        pipeline.fit(x_train, y_train)
        # Accuracy
        train_score = accuracy_score(y_train, pipeline.predict(x_train), normalize=True, s
ample_weight=None)
        test_score = accuracy_score(y_test, pipeline.predict(x_test), normalize=True, samp
le_weight=None)
        df = pd.DataFrame({"max_depth" : [max_depth], "n_estimators" : [n_estimators], "tr
ain_score" : [train_score], "test_score" : [test_score]})
        result = result.append(df, ignore_index=True)

result.sort_values("test_score", ascending=False).head()
```

Out[101]:

	max_depth	n_estimators	train_score	test_score
599	74.0	65.0	0.964824	0.760234
922	82.0	60.0	0.964824	0.760234
839	80.0	59.0	0.964824	0.760234
786	79.0	47.0	0.964824	0.754386
1050	85.0	65.0	0.964824	0.754386

Decision Tree Classifier Searching Best coefficients

In [98]:

```
# Find best coefficients for DecisionTreeClassifier
result = pd.DataFrame({"max_depth" : [], "train_score" : [], "test_score" : []})
for max_depth in range(5, 101):
    pipeline = Pipeline([
        ('preprocessing', column_transformer),
        ('classifier', DecisionTreeClassifier(max_depth=max_depth))
    ])
    pipeline.fit(x_train, y_train)
    # Accuracy
    train_score = accuracy_score(y_train, pipeline.predict(x_train), normalize=True, sampl
e_weight=None)
    test_score = accuracy_score(y_test, pipeline.predict(x_test), normalize=True, sample_w
eight=None)
    df = pd.DataFrame({"max_depth" : [max_depth], "train_score" : [train_score], "test_sco
re" : [test_score]})
    result = result.append(df, ignore_index=True)

result.sort_values("test_score", ascending=False).head()
```

Out[98]:

	max_depth	train_score	test_score
86	91.0	0.972362	0.701754
91	96.0	0.972362	0.701754
44	49.0	0.972362	0.701754
35	40.0	0.972362	0.695906
73	78.0	0.972362	0.695906

K-Neighbors Classifier Searching Best coefficients

In [101]:

```
# Find best coefficients for KNeighborsClassifier
result = pd.DataFrame({"n_neighbors" : [], "train_score" : [], "test_score" : []})
for n_neighbors in range(5, 80):
    pipeline = Pipeline([
        ('preprocessing', column_transformer),
        ('classifier', KNeighborsClassifier(n_neighbors=n_neighbors))
    ])
    pipeline.fit(x_train, y_train)
    # Accuracy
    train_score = accuracy_score(y_train, pipeline.predict(x_train), normalize=True, sampl
e_weight=None)
    test_score = accuracy_score(y_test, pipeline.predict(x_test), normalize=True, sample_w
eight=None)
    df = pd.DataFrame({"n_neighbors" : [n_neighbors], "train_score" : [train_score], "test
_score" : [test_score]})
    result = result.append(df, ignore_index=True)

result.sort_values("test_score", ascending=False).head()
```

Out[101]:

	n_neighbors	train_score	test_score
21	26.0	0.665829	0.637427
20	25.0	0.658291	0.631579
1	6.0	0.736181	0.631579
16	21.0	0.658291	0.625731
2	7.0	0.726131	0.625731