MCCOY COLLEGE OF BUSINESS, TEXAS STATE UNIVERSITY

Course: **QMST5336 ANALYTICS**

Assignment 2
Answer Example

Instructor: *Dr. Emily Zhu*

Due date: *April 4th, 2021*

Please note that the answers/methods are NOT unique. The provided codes are for reference ONLY.

1. The God, The Bad and The Ugly: Ratings for Austin *Airbnb* Properties

We now aim to identify the good, the soso and the bad properties by studying six(6) review scores for accuracy, cleanliness, checkin, communication, location and value, of those listings with available rating records, in *reviews.csv* (in-sample/training data). By deriving the evaluating criteria from these existing listings, please classify 28 ratings in *Reviews4Cluster.h5* (out-of-sample/testing set) into 3 clusters of good, medium and bad listings, respectively.

- Use hierarchical clustering to classify properties regarding the given six review scores. What problem(s) do you have when running Python? Please try to draw a random sample of 150 properties and run hierarchical clustering on the smaller sample.
- Use K-means to cluster properties regarding the given six review scores. Compare the outcomes from hierarchical and K-Means clustering. Are the clustering results the same under different clustering methods?
- Will the clustering above be the same if replacing missing values with modes of the corresponding review scores?
  Hint: You may make a table to illustrate the comparisons.
- Do you think it reasonable to classify those ratings into 3 categories?
  Hint: You may want to check Within-Cluster-Sum-of-Squares (WCSS) in K-Means.

Answer. We list the answers to the first two questions in a table for the purpose of comparison:

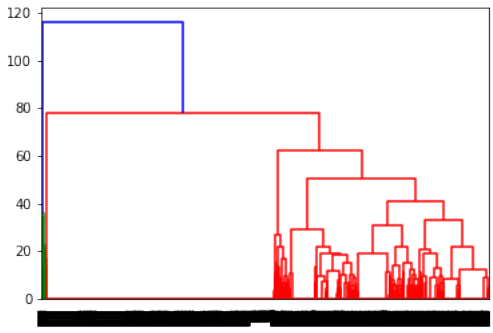
	Clustering			
ID*	kmean**		hierarchy	
	delete NAN	replace NAN	delete NAN	replace NAN
1	1	1	0	0
2	2	1	0	0
3	1	2	0	0
4	0	0	0	0
5	1	1	0	0
6	1	1	1	1
7	0	1	2	2
8	2	1	0	0
9	1	2	0	0
10	0	0	0	0
11	1	1	0	0
12	1	1	1	1
13	0	1	2	2
14	2	2	0	0
15	2	0	0	0
16	2	1	0	0
17	2	1	1	1
18	2	1	2	2
19	0	0	0	0
20	1	2	0	0
21	1	2	1	1
22	0	2	2	2
23	0	0	0	0
24	0	0	1	1
25	0	0	2	2
26	1	1	0	0
27	0	1	2	2
28	0	1	1	1

∗The ID refers to the new review ratings in *Reviews4Cluster.h5*.
∗∗ K-Means is not a deterministic algorithm, in this case, you may end up with different prediction results.

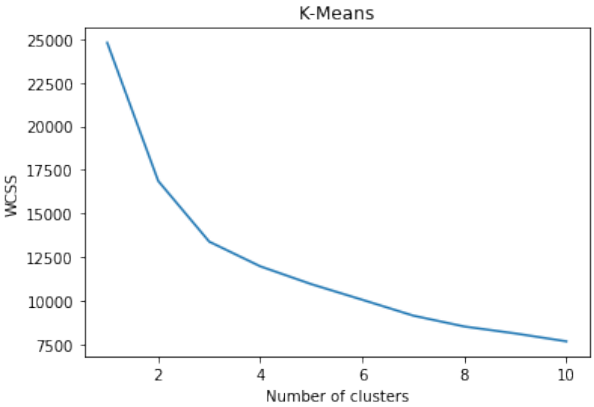
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering

#---Historical Review Ratings--------------
df_a = pd.read_csv('reviews.csv', usecols = ['review_scores_accuracy',
    'review_scores_cleanliness', 'review_scores_checkin', '
    review_scores_communication', 'review_scores_location', '
    review_scores_value'])
df_b = df_a.dropna(axis=0) #drop NaN for the 1st question
df_b = df_a.fillna(df_a.mode()) # replace missing values with the modes
    for the 2nd question

#---Retrieve New Ratings for Prediction-----------
```

```
data_store = pd.HDFStore('Reviews4Cluster.h5') # Retrieve data using key
df_new = data_store['df_review_cluster']
data_store.close()

#----Apply K-Means Clustering-------------
kmeans = KMeans(n_clusters = 3)
# fit kmeans object to data
kmeans.fit(df_b)
# print location of clusters learned by kmeans object
print(kmeans.cluster_centers_)
# save new clusters for chart
y_km = kmeans.fit_predict(df_new)

#----Apply Hierarchy Clustering---------------
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
#You should plot dendrogram first***
dendrogram = sch.dendrogram(sch.linkage(df_b, method='ward'))
hc = AgglomerativeClustering(n_clusters=3, affinity = 'euclidean',
    linkage = 'ward')
y_hc = hc.fit_predict(df_new)

#----***You May Need this: Expand the limit for recursion-----------
import sys
sys.setrecursionlimit(10000)
#DO REMEMBER to set back to default after running Hierarchy Clustering
sys.setrecursionlimit(1000)
```

The dendrogram may look like



The number of clusters is 3, which is reasonable as Within-Cluster-Sum-of-Squares (WCSS) in K-Means shows below



```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state =
   42)
    kmeans.fit(df_b)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
```

No Traffic Jams In the Extra Mile:
Weather, Traffic & Planning

Next, we study the traffic volumes, how weather affects traffic and temperature prediction based on the file
Metro_Interstate_Traffic_AMPeak_cleaned.csv, an extract from the original dataset[1].

2. How Does Weather Affect Traffic Volumes?

We focus on traffic volumes in rush hours and take 8AM for instance. In other words, the sample of our interest consists of observations at 8AM ONLY (one observation per day from 2015 to 2018). We would like to examine how the following factors affect the *traffic volumes*.

- *weekend*, including Saturdays, Sundays and holidays in the column *holiday*;
- *temperature* as shown in the column *temp*;
- *amount of clouds* as shown in the column *clouds_all*;
- *spot snowfall* as shown in the column *snow_1h*.

Please compare the following two models, M1 & *M2*, in terms of statistical significance (R^2, adjusted R^2) and predicting power (out-of-sample RMSE). You can divide the sample into 80% training and 20% testing sets. Which model would you recommend and why?
(Hint: We do NOT need to consider the time series in this case, because we do NOT study any association with time.)

traffic volume $= b_0 + b_1(weekend) + b_2(clouds) + b_3(temperature) + b_4(snowfall)$ $(M1)$;
$trafficvolume = b_0 + b_1(weekend) + b_2(clouds) + b_3(snowfall)$ $(M2)$.

Answer. We prefer M2 as it is more statistically significant (higher R square and Adjusted R square) and stronger prediction power (lower RMSE).

	M1	M2
R square	.756	.738
Adjusted R	.755	.737
RMSE	830.04	839.83

Please note that the answers are NOT unique, as the test and training sets are split at random.

[1]Data source and descriptions of the data can be found at `https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume`

```
import pandas as pd
import numpy as np

df_clean = pd.read_csv('Metro_Interstate_Traffic_cleaned.csv',
    parse_dates = True, encoding = "utf_8_sig")

#-----Regression Model 1-----------
import statsmodels.formula.api as smf
# create a fitted model
lm1 = smf.ols(formula = 'traffic_volume ~ weekend + temp + clouds_all +
    rain_1h + snow_1h', data = df_slt).fit()
lm1.summary()

#----Regression Model 2--------------
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
# Out-of-Sample Evaluation
X = df_slt[['weekend', 'temp', 'clouds_all', 'rain_1h', 'snow_1h']]
y = df_slt.traffic_volume

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
    0.2, random_state=1)

# Instantiate model
lm2 = LinearRegression()

# Fit Model
lm2.fit(X_train, y_train)

# Predict
y_pred = lm2.predict(X_test)

# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

3. Can We Predict Average Weekly Temperature?

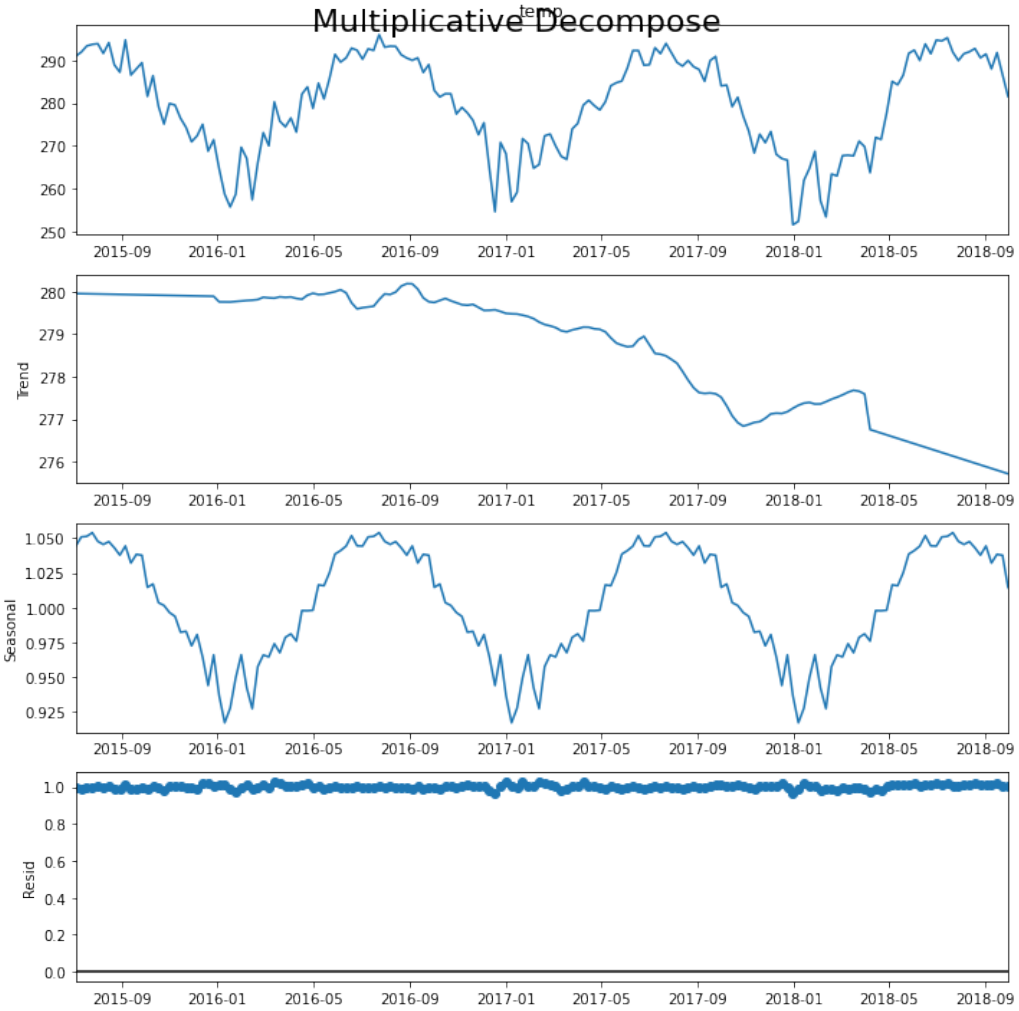
Resample the variable temperature *Temp* into weekly basis; and

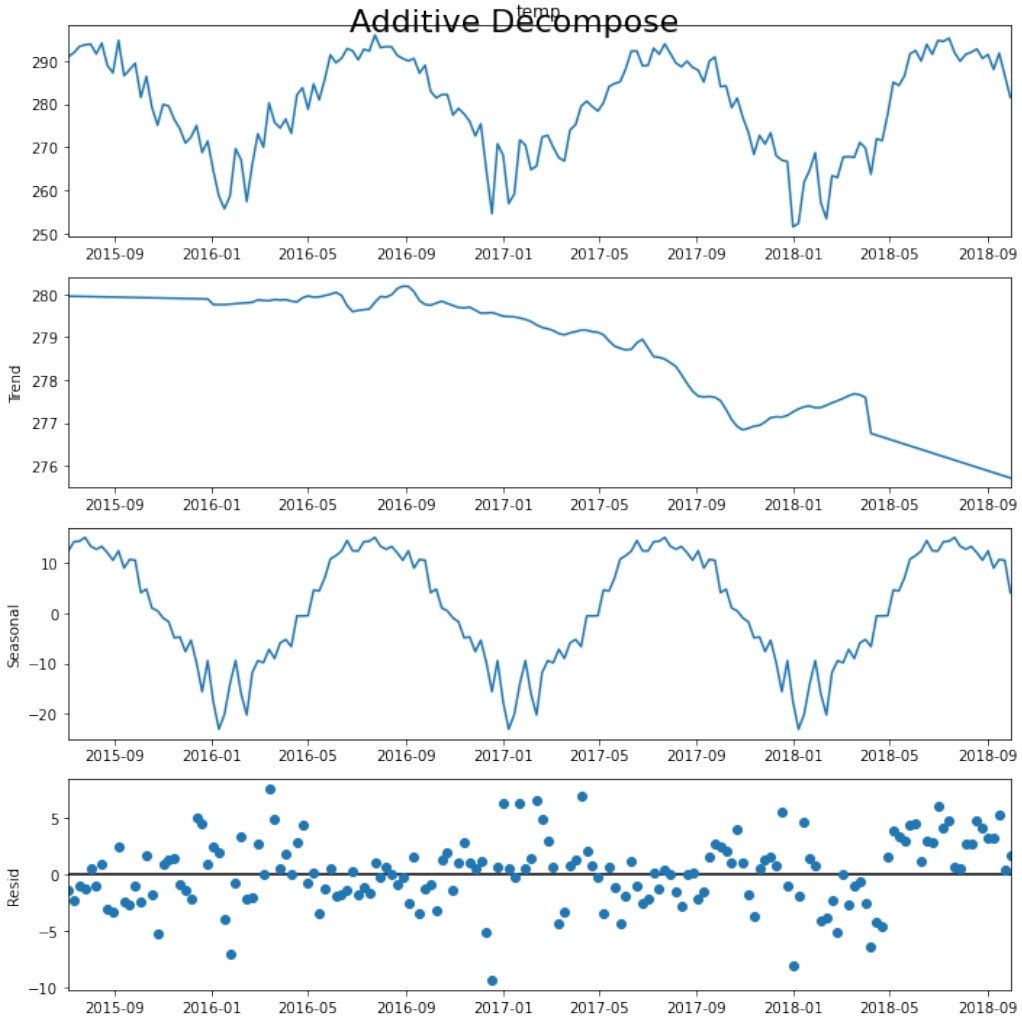
a. decompose the time series *Temp* and which decomposition (additive or multiplicative) is appropriate?

b. cross validate RMSE on the later 30% of the time series.

c. predict average weekly temperature for the next year starting on Oct. 7, 2018.

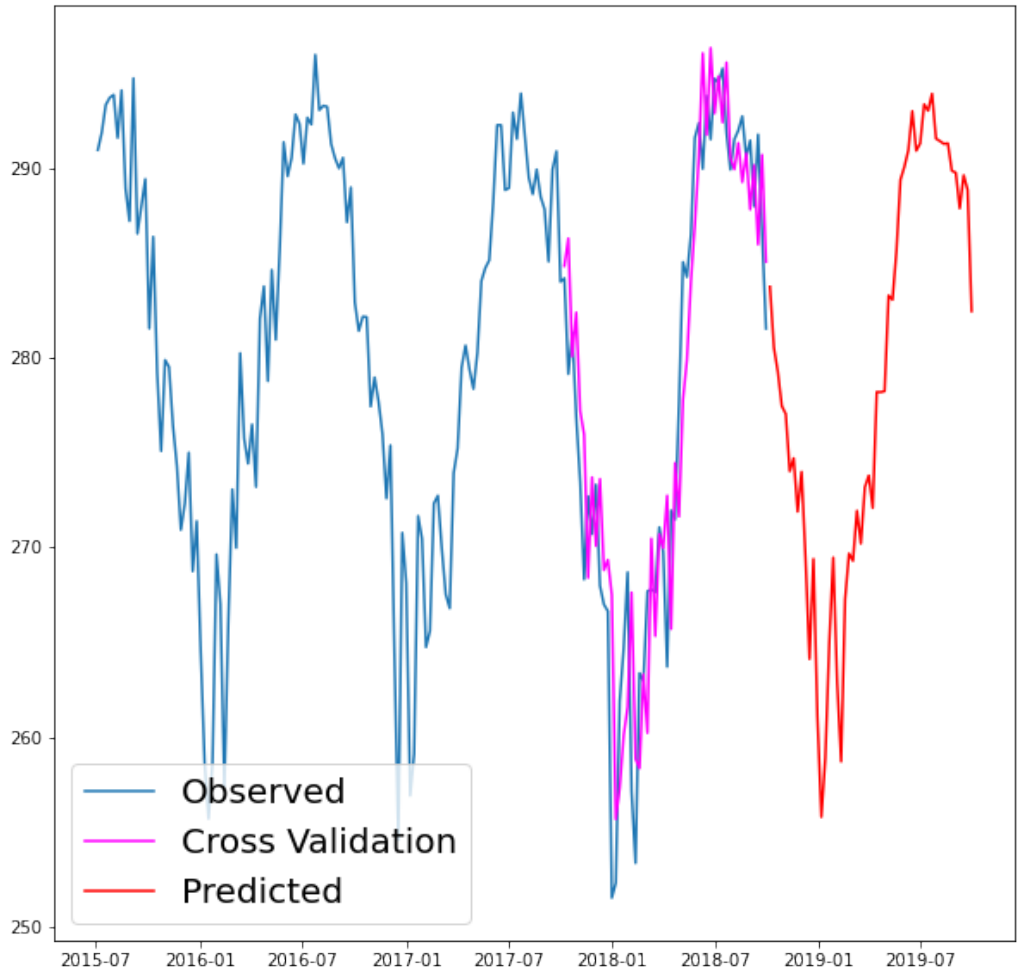
Answer.


Multiplicative Decompose
temp
Trend
Seasonal
Resid
2015-09
2016-01
2016-05
2016-09
2017-01
2017-05
2017-09
2018-01
2018-05
2018-09



Because the residuals of additive decomposition are randomly scattered, it is an appropriate decomposition.

RMSE of cross validation is 4.965452802567684 degrees.




```
df1 = df_full[['temp', 'date']]
df1.set_index('date, inplace = True)
df1.index = df1.index.astype('datetime64[ns]')
df = df1.resample('W', label='right', closed = 'right').mean()

##-----Decompose----
from matplotlib import pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
# Multiplicative Decomposition
result_mul = seasonal_decompose(df['temp'], model= 'multiplicative',
    extrapolate_trend= 'freq')
#Setting extrapolate_trend='freq' takes care of any missing values in
    the trend and residuals at the beginning of the series.
# Additive Decomposition
result_add = seasonal_decompose(df['temp'], model= 'additive',
    extrapolate_trend= 'freq')
# Plot
plt.rcParams.update({'figure.figsize': (10,10)})
result_mul.plot().suptitle('Multiplicative Decompose', fontsize=22)
result_add.plot().suptitle('Additive Decompose', fontsize=22)
plt.show()

##----Predict trend, seasonality and residual separately-------
from statsmodels.tsa.ar_model import AutoReg, ar_select_order
from sklearn.metrics import mean_squared_error
series = df['temp']
```

```
historic = series.iloc[:int(len(series) * 0.7)]
test = series.iloc[int(len(series) * 0.7):]
historic = historic.to_list()
prediction = []

for i in range(len(test)):
    sel = ar_select_order(historic, 13, old_names=False)
    sel.ar_lags
    model_fit = sel.model.fit()
    pred = model_fit.predict(start=len(historic), end=len(historic),
   dynamic=False)
    predictions.append(pred[0])
    historic.append(test[i])
cross_val = pd.Series(predictions, index=test.index)

test_score = mean_squared_error(test, cross_val, squared = False)
print(test_score)

##----Prediction for the next year----
#existing data
historic = series
#prediction for the next 52 weeks (1 year)
n_pred = 52
#the future weeks starts on 2018-09-30, because the last existing data
   ends on 2018-10-07
date_pred = pd.date_range("2018-10-07", periods=n_pred, freq="w")

sel = ar_select_order(historic, 13, glob = True, seasonal = True,
   old_names=False)
sel.ar_lags
model_fit = sel.model.fit()
pred = model_fit.predict(start=len(historic), end=len(historic) + n_pred
    - 1, dynamic=False)
predictions = pd.Series(pred.values, index=date_pred)

# plot results
plt.plot(series, label='Observed')
plt.plot(cross_val, color='magenta', label='Cross Validation')
plt.plot(predictions, color='red', label='Predicted')
plt.legend(fontsize = 20)
plt.show()
```