Buckman Internship Screening Test Report

Aswath Rao V S,

16PD05,

MSc Data Science,

PSG College of Technology

Coimbatore

Q1.Prediction problem

Problem Statement –

The dataset (Dataset Problem 1) is provided for the prediction of the noise pressure level. The data set has the following attributes
1) Frequency (Hz)
2) Angle (Degrees)
3) Chord Length (m)
4) Velocity – Free-stream velocity(m/s)
5) Displacement – Suction side displacement thickness (m)

You are required to predict the generated noise i.e., Noise Pressure (Decibels) Analyse the patterns and insights from the following dataset and build a model which has the best accuracy for prediction. Explain all the steps from data pre-processing, model selection, model validation etc.

Objective

To predict the values of generated Noise.

Understanding the problem

Since output is to be predicted from a set of inputs, Supervised learning algorithm must be used.

Exploratory Data Analysis

1. data.describe()

	Frquency(Hz)	Angle	Chord_Length	velocity	Displacement	Pressure_level
count	1503.000000	1503.000000	1503.000000	1503.000000	1503.000000	1503.000000
mean	2886.380572	6.782302	0.136548	50.860745	0.011140	124.835943
std	3152.573137	5.918128	0.093541	15.572784	0.013150	6.898657
min	200.000000	0.000000	0.025400	31.700000	0.000401	103.380000
25%	800.000000	2.000000	0.050800	39.600000	0.002535	120.191000
50%	1600.000000	5.400000	0.101600	39.600000	0.004957	125.721000
75%	4000.000000	9.900000	0.228600	71.300000	0.015576	129.995500
max	20000.000000	22.200000	0.304800	71.300000	0.058411	140.987000

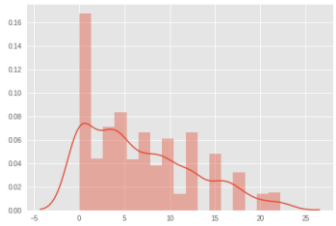
2. To check whether is there any null value in the data :

```
df.describe() #No null values
datasetHasNan = False
if df.count().min() == df.shape[0]:
    print('NO null values')
else:
    datasetHasNan = True
    print('Null is present')
```

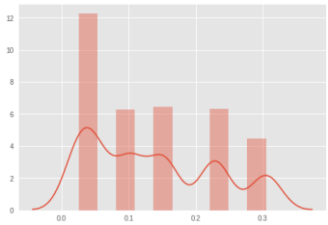
Inference : No null

3. Distribution of features

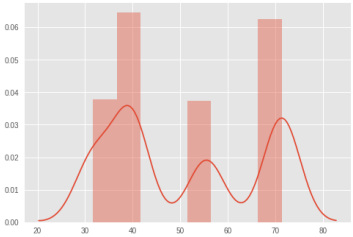
```
f1 = [df['Angle'].values]
sns.distplot(f1)
```



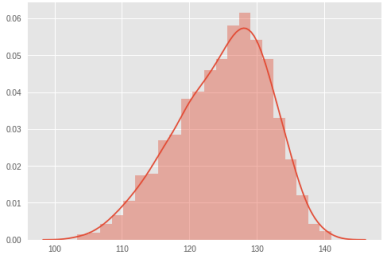
```
f2 = [df['Chord_Length'].values]
sns.distplot(f2)
```



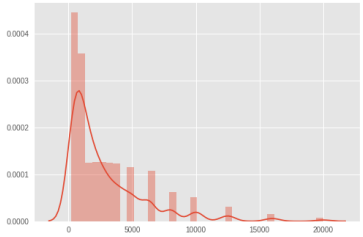
```
f2 = [df['velocity'].values]
sns.distplot(f2)
```



```
f4 = [df['Pressure_level'].values]
sns.distplot(f4)
```

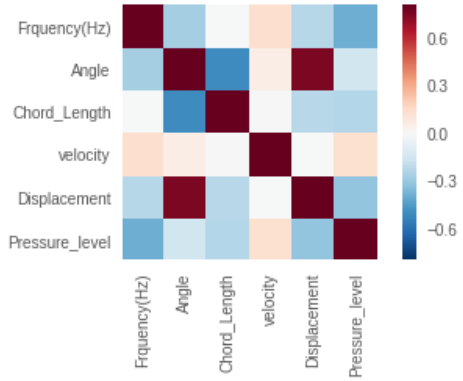


```
f5 = [df['Frquency(Hz)'].values]
sns.distplot(f5)
```



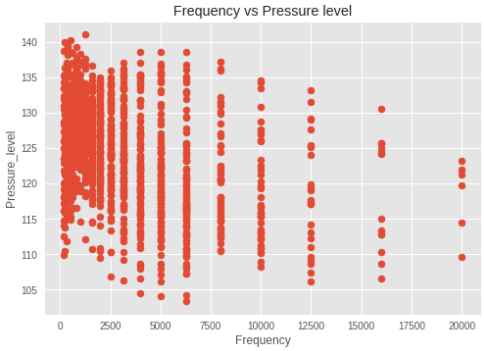
4.Corelation matrix

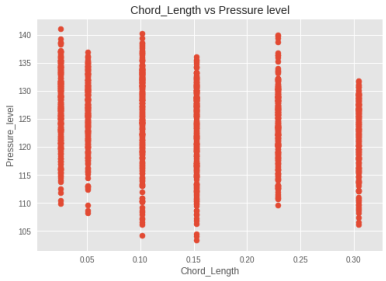
```
correlation_matrix = df.corr()
fig = plt.figure(figsize=(6,3))
sns.heatmap(correlation_matrix,vmax=0.8,square = True)
plt.show()
```

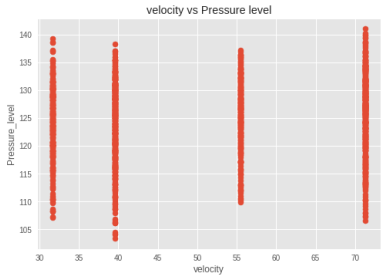


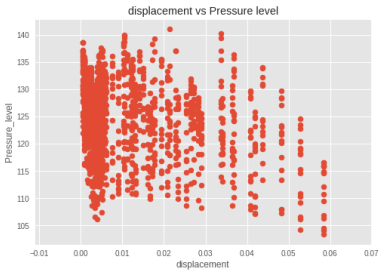
	Frquency(Hz)	Angle	Chord_Length	velocity	Displacement	Pressure_level
Frquency(Hz)	1.000000	-0.272765	-0.003661	0.133664	-0.230107	-0.390711
Angle	-0.272765	1.000000	-0.504868	0.058760	0.753394	-0.156108
Chord_Length	-0.003661	-0.504868	1.000000	0.003787	-0.220842	-0.236162
velocity	0.133664	0.058760	0.003787	1.000000	-0.003974	0.125103
Displacement	-0.230107	0.753394	-0.220842	-0.003974	1.000000	-0.312670
Pressure_level	-0.390711	-0.156108	-0.236162	0.125103	-0.312670	1.000000

5.Scatter plot between pair of features

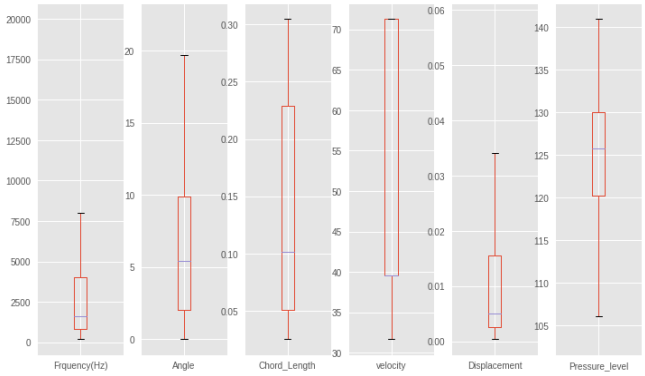








6. Outlier detection



Inference - There are no outliers

7.Splitting the data

Train set and test set are separated by hold out method

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=12345)
```

8.Model building

Approach 1 :

Multiple Linear Regression

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
print("Coefficient ", regressor.coef_)
print("Intercept ",regressor.intercept_)
```

Model fitted :

```
Coefficient  [-3.93510402 -1.98234861 -3.00501798  1.45058158 -1.67180338]
Intercept  124.81742395437261
```

```
y_pred = regressor.predict(X_test)
```

Mean Absolute Error :

```
sum(abs(list(y_test)-y_pred))/len(y_pred)
```

Mean Absolute Errror :

```
3.8426679451932415
```

R-square value :

```
regressor.score(X_train,y_train)
```

```
0.5281887734416073
```

Approach 2 :

Using ensemble method

- Random forest

```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf.fit(X_train,y_train);
```

```
# Use the forest's predict method on the test data
predictions = rf.predict(X_test)
# Calculate the absolute errors
errors = abs(predictions - y_test)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 2.54 degrees.
```

In terms of Error , Random forest has given better results

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / y_test)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

Accuracy achieved : 97.96%

To find the best parameter using grid Search and using that we do random forest

```
from sklearn.model_selection import cross_val_predict, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
gsc = GridSearchCV(
        estimator=RandomForestRegressor(),
        param_grid={
            'max_depth': range(3,7),
            'n_estimators': (10, 50, 100, 1000),
        },
        cv=5, scoring='neg_mean_squared_error', verbose=0,                    n_jobs=-1)
```

```
grid_result = gsc.fit(X_train, y_train)
```

```
best_params = grid_result.best_params_
```

```
rfr = RandomForestRegressor(max_depth=best_params["max_depth"], n_estimators=best_params["n_estimators"],
```

```
scores = cross_val_predict(rfr, X_train, y_train, cv=10)
```

Results -

```
errors = abs(predictions - y_test)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 2.99 degrees.
```

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / y_test)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 97.59 %.
```

Q2.

Understanding the problem:

Given the 3 datasets which consists the details of Customers

1. Customer_Demographics.xlsx – consists details of Customers
2. Customer_Transaction.xlsx – consists Transaction details
3. Store_Master.xlsx – consists details of the stores present

Data is imported in oracle

Query 1 :

Write a SQL query to find out those customers and their demographics who have visited a store for more than ten times.

Solution:

Group by, count and Joins are used to solve this problem

Query:

```
select a.*,b.Store_Code from Customer_Transaction a, (select
count(*),Customer_ID, Store_Code from Customer_Transaction group by
(Store_Code, Customer_ID) having count(*) >=10) b where
a.Customer_ID=b.Customer_ID
```

Sample Output:

ID	TERRITORY	BUSINESS	YEAR	WEEK	STORE_CODE	CITY_NAME	STORE_TYPE	TRANSACTION_TYPE	RETURN_REASON	CUSTOMER_ID	INVOICES	ITEM_COUNT	REVENUE	DISCOUNT	UNITS_SOLD
1	United Arab Emirates	Max	2010	0	60065	Dubai	Stand alone	Return	Size Problem	1800000058056860	1	3	-169	0	-3
2	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000020917140	1	1	0	0	0
3	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000021265010	1	1	0	0	0
8	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000046079170	2	7	0	0	0
10	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000048130990	1	2	0	0	0
14	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000058101370	1	1	0	0	0
15	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000058661850	1	1	0	0	0
16	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000080131780	1	1	0	0	0
17	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000080464400	1	2	0	0	0
19	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000117190630	1	1	0	0	0
19	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000117190630	1	1	0	0	0
19	United Arab Emirates	Max	2016	0	60053	-	-	Purchase	-	1800000117190630	1	1	0	0	0

Query 2 :

Write a SQL query to create a column called “customer rank” which will rank the customers based on their frequent visits.

Solution :

Group by, count and Rank are used to solve this problem

Query :

select Customer_ID, RANK() OVER(ORDER BY freq desc) as rank from (select Customer_ID, count(*) as freq from Customer_Transaction group by Customer_ID order by count(*) desc)

Sample Output :

CUSTOMER_ID	RANK
1800000047344470	1
1800000039107220	2
1800000054611290	3
1800000068164060	4
1800000045739070	5
1800000045625720	5
1800000028545190	7
1800000011640790	7
1800000252214780	9
1800000007483180	10
1800000021609980	11
1800000117348150	12
1800000020882190	13
1800000029493990	14
1800000167357240	15
1800000106776580	15
1800000054767430	15
1800000166642220	18
1800000046555510	19
1800000008067550	20
1800000105728840	21
1800000007209620	21
1800000183441690	21
1800000007444410	24
1800000029723590	24
1800000088072080	26
1800000054639940	27
1800000057787330	28

Query 3 :

Write a SQL Query which will create a master table (Combined all the three tables) for all the stores and customers taken into account.

Solution :

Inner Joins are used to solve this problem

Query :

```
SELECT Customer_Demographics.Customer_ID, Customer_Transaction
.Store_Code
FROM ((Customer_Demographics
INNER JOIN FIRST ON Customer_Demographics.Customer_ID =
Customer_Transaction.Customer_ID)
INNER JOIN  Store_Master  ON Customer_Transaction.Store_Code =
Store_Master.Store_Code);
```

Sample Output:

CUSTOMER_ID	STORE_CODE
1800000058056860	60065
1800000020917140	60053
1800000021265010	60053
1800000025693950	60053
1800000044104620	60053
1800000045703150	60053
1800000045740150	60053
1800000046079170	60053
1800000047493340	60053
1800000048130990	60053
1800000054019360	60053
1800000054841260	60053
1800000054994790	60053
1800000058101370	60053
1800000058661850	60053
1800000080131780	60053
1800000080464400	60053
1800000089762380	60053
1800000117238230	60053
1800000134189220	60053
1800000134965000	60053
1800000142189030	60053
1800000142779000	60053
1800000143039020	60053
1800000143403790	60053
1800000184953240	60053
1800000006395940	60053
1800000006481140	60053

Q3. Time Series Analysis

Understanding the data set :

The data set Dataset Problem 3.xlsx consists of Time series data of power production from 2006-2017. The data set has 5 attributes with Date, Power, Consumption and Energy

Inference from data set

The data set consists of missing values for columns E1,E2 and E3 where missing values of E1 is lesser

We have complete data from 2012

Steps followed :

1. Question given
2. Understanding the problem
3. Approach
4. Code Snippet
5. Graph

Question given:

Plot the same graph given below for 2017 instead of the consumption rate take E1, E2 in the Y-axis (Note: Separate graph for E1 & E2 with the months in a year as X-axis)

Understanding the problem:

Plot the graph as given below by taking E1 for the year 2017
Plot the graph as given below by taking E2 for the year 2017

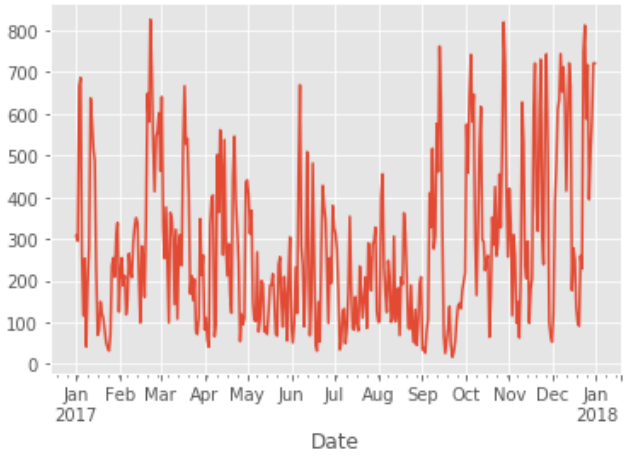
Approach:

The year 2017 is taken from Timestamp pandas ,Date is set as the Index and graph is plotted for E1 and E2

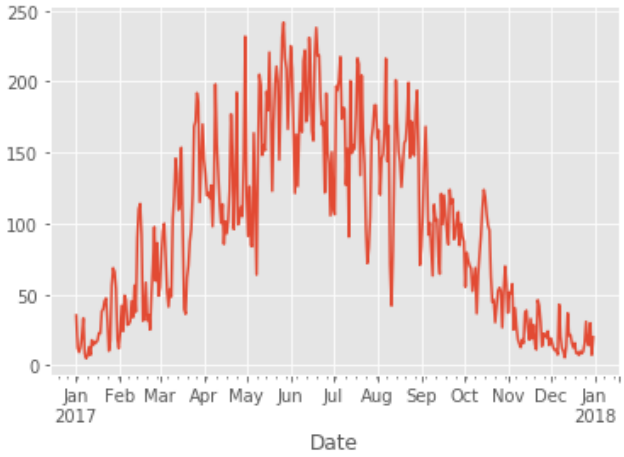
Code Snippet:

```
df = df[df.Date >= pd.Timestamp(2017,1,1,0)]
df = df.set_index('Date',drop="True")
df['E2'].plot()
df['E1'].plot()
```

Graph for E1:



Graph for E2:



Question given:

Aggregate the data on a seasonal basis for E1 and E2 and plot the same for the year 2017.

Understanding the problem:

To Infer the seasonal trend in 2017

Approach:

The data in 2017 is aggregated into 4 season and trend is analyzed using the given plot

Code Snippet :

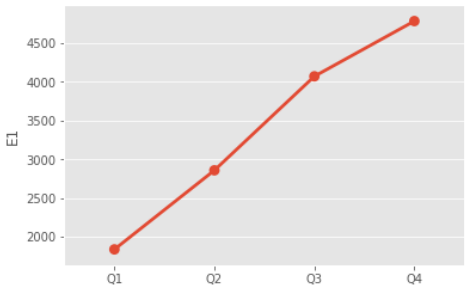
```
s =[]
w =[]
y =[]
for i in range(1,13):
    s.append(np.sum(list(df[df.index.month == i]['Power Consumption'])))
    w.append(np.sum(list(df[df.index.month == i]['E2'])))
    y.append(np.sum(list(df[df.index.month == i]['E1'])))

a =[]
b = []
c =[]
for i in range(int(len(s)/3)):
    a.append((np.mean(s[i:i+3])))
    b.append((np.mean(w[i:i+3])))
    c.append((np.mean(y[i:i+3])))

import seaborn as sns
Q = ['Q1','Q2','Q3','Q4']
sns.pointplot(x=Q,y=df['E2'])

Q = ['Q1','Q2','Q3','Q4']
sns.pointplot(x=Q,y=df['E1'])
```

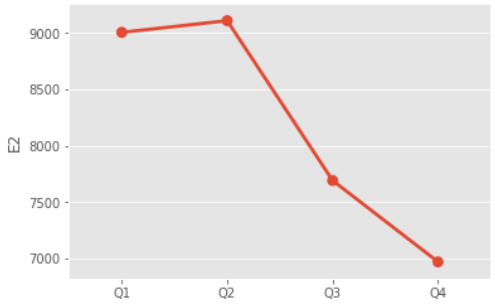
Graph for E1:



Observation:

There is an Increase trend for E1

Graph for E2:



Observation:

There is an Increase trend for E2

Question given :

Does the power consumption depend on E1 and/or E2? Justify your answer using plots/tables?

Understanding the problem :

To find correlation between power and E1,E2

Approach:

Correlation matrix is tabulated

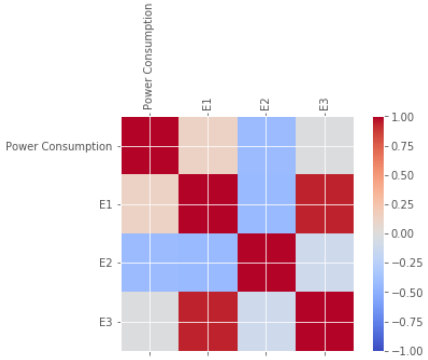
Code Snippet :

```
df.corr()
```

Table:

	Power Consumption	E1	E2	E3
Power Consumption	1.000000	0.125619	-0.416705	-0.013310
E1	0.125619	1.000000	-0.425259	0.944424
E2	-0.416705	-0.425259	1.000000	-0.104101
E3	-0.013310	0.944424	-0.104101	1.000000

Graph:



- E2 is more depended on Power
- E2 and Power are negatively co-related

Q4. Problem on Statistics

Given problem –

From customer transaction data provided in dataset, find the probability of each customer visiting each store. Say, for a given customer what is the probability that he will visit a store? Include your output as probability.csv and explain the formula.

Insights from Data:

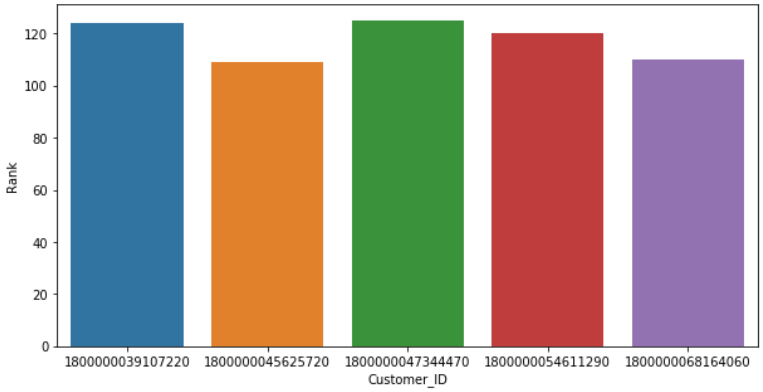
There are 1,00,000 Unique Customers and 32 Unique Stores.
Majority of the customers haven't visited multiple stores.

Solution:

Taking the frequency of visit to a store by a particular customer, the probability distributed table is generated. Say a customer visits 3 out of 32 stores with frequencies 10,23,18 the probability of the customer visiting it again would be 10/51, 23/51 and 18/51 respectively.
So the entire space (100000X32) is calculated as mentioned above.

P(visiting a store) = Frequency he visited that store / Total visits made to all the stores

Top 5 Customers with their ranks



Coding Approach 1

Every permutation of customer, store is generated to find the frequency and followed probability. This will cost O(100000x32) . This is performed using nested loop.

Coding Approach 2

Customer and stores are grouped first and then the group object is iterated to find frequency of visits made. This reduced to O(1,84,000)

Coding snippet :

```
c = 0
visit = []

for customer in unique_customers:
    visit_counts = []

    for store in unique_stores:
        temp = df[(df['Store_Code'] == store) & (df['Customer_ID'] == customer)]
        visit_counts.append(temp.shape[0])

    total = sum(visit_counts)
    visit_counts[:] = [x / total for x in visit_counts]
    c = c+1
    visit.append(visit_counts)
    if c%100 ==0:
      print(c)
```

Sample Output :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	0.958333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0.823529	0	0	0	0	0	0	0	0	0	0	0	0.058824	0	0.117647	0	0	0	0	0	0
2	0	0.807018	0	0	0	0	0	0	0	0	0	0	0	0.140351	0	0.017544	0	0	0	0	0	0
3	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0.235294	0.176471	0.058824	0	0	0	0	0	0	0	0	0	0	0.117647	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0.103896	0.818182	0	0	0.012987	0	0	0	0	0	0	0	0	0	0.051948	0	0	0	0	0	0
8	0	0.777778	0	0	0	0	0	0	0	0	0	0	0	0.111111	0	0.111111	0	0	0	0	0	0
9	0	0.8	0	0	0	0	0	0	0	0	0	0.066667	0	0	0	0	0	0	0	0	0	0
10	0	0.347826	0.043478	0	0	0	0	0	0	0	0	0	0	0.304348	0	0.304348	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0.625	0	0	0	0	0	0	0	0	0	0	0	0.125	0	0.25	0	0	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0.833333	0	0	0	0	0	0	0	0	0	0	0	0	0	0.166667	0	0	0	0	0	0
15	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0
16	0	0.157895	0.236842	0	0	0.052632	0	0	0	0	0	0	0	0.263158	0	0.184211	0	0	0	0	0	0
17	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0.375	0	0	0	0	0	0	0	0	0	0	0	0.3125	0	0.3125	0	0	0	0	0	0
19	0	0.352941	0	0	0	0.588235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0.428571	0	0	0	0	0	0	0	0	0	0	0	0.333333	0	0.238095	0	0	0	0	0	0
21	0	0.857143	0	0	0	0	0	0	0	0.071429	0	0	0	0	0	0.071429	0	0	0	0	0	0
22	0	0.823529	0	0	0	0	0	0	0	0	0	0	0	0.176471	0	0	0	0	0	0	0	0
23	0	0.438356	0.013699	0	0	0	0	0	0	0	0	0	0	0.232877	0	0	0	0	0	0	0	0
24	0	0.764706	0.058824	0	0	0	0	0	0	0.058824	0	0	0	0.117647	0	0	0	0	0	0	0	0
25	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0