In [1]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("exchange-rates-2000-2022.csv")
df
```

Out[1]:

	DATE	USD	GBP	EUR	JPY100	CHF	AUD	CAD	SGD	HKD100
0	3-Jan-00	3.8	6.1598	3.8646	3.7259	2.4066	2.5002	2.6288	2.2885	48.8715
1	4-Jan-00	3.8	6.2113	3.8956	3.7079	2.4262	2.5021	2.6257	2.2923	48.863
2	5-Jan-00	3.8	6.2246	3.911	3.6936	2.4382	2.4915	2.615	2.2945	48.8579
3	6-Jan-00	3.8	6.2466	3.9239	3.6442	2.4453	2.5002	2.6206	2.2948	48.8586
4	7-Jan-00	3.8	6.2603	3.9186	3.6087	2.4417	2.4854	2.6043	2.2843	48.8535
...	...	...	...	...	...	...	...	...	...	...
17012	23-Dec-22	0.0187	0.1073	2.78	0.2115	5.3486	120.569	1.9574	3.343	0.1789
17013	27-Dec-22	0.0188	0.1072	2.7768	0.2112	5.3442	120.3763	1.953	3.3403	0.1786
17014	28-Dec-22	0.0188	0.1074	2.7789	0.2117	5.3502	120.6643	1.9557	3.3439	0.1794
17015	29-Dec-22	0.0187	0.107	2.799	0.2111	5.3399	120.3169	1.9488	3.3374	0.1784
17016	30-Dec-22	0.0187	0.1069	2.7899	0.2108	5.3342	120.1601	1.9466	3.3338	0.1783

In [2]:
```
#A. Explanatory Data Analysis (EDA)
```

In [3]:
```
df.shape
```

Out[3]:
```
(17017, 10)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['DATE', 'USD', 'GBP', 'EUR', 'JPY100', 'CHF', 'AUD', 'CAD', 'SGD',
       'HKD100'],
      dtype='object')
```

In [5]:
```
df.columns = map(str.upper, df)
df.rename(columns=lambda x:x+'_MYR', inplace=True)
df.rename(columns={'DATE_MYR':'Date'}, inplace=True)
```

In [6]:
```
1 df.head(15)
```

Out[6]:

	Date	USD_MYR	GBP_MYR	EUR_MYR	JPY100_MYR	CHF_MYR	AUD_MYR	CAD_MYR	SGD_
0	3-Jan-00	3.8	6.1598	3.8646	3.7259	2.4066	2.5002	2.6288	2
1	4-Jan-00	3.8	6.2113	3.8956	3.7079	2.4262	2.5021	2.6257	2
2	5-Jan-00	3.8	6.2246	3.911	3.6936	2.4382	2.4915	2.615	2
3	6-Jan-00	3.8	6.2466	3.9239	3.6442	2.4453	2.5002	2.6206	2
4	7-Jan-00	3.8	6.2603	3.9186	3.6087	2.4417	2.4854	2.6043	2
5	11-Jan-00	3.8	6.2227	3.9007	3.6132	2.4225	2.4909	2.6107	2
6	12-Jan-00	3.8	6.2641	3.9265	3.588	2.4381	2.5017	2.6088	2
7	13-Jan-00	3.8	6.2569	3.9136	3.6002	2.4294	2.5175	2.6146	2
8	14-Jan-00	3.8	6.2554	3.8999	3.5798	2.4184	2.5314	2.6201	2
9	17-Jan-00	3.8	6.2183	3.8509	3.6087	2.3868	2.528	2.6185	2
10	18-Jan-00	3.8	6.2094	3.8384	3.6201	2.378	2.5211	2.623	2
11	19-Jan-00	3.8	6.2238	3.8559	3.5978	2.3904	2.5245	2.6195	
12	20-Jan-00	3.8	6.2468	3.8429	3.6081	2.3813	2.5207	2.6166	2
13	21-Jan-00	3.8	6.2805	3.8623	3.6208	2.3967	2.5291	2.6282	2
14	24-Jan-00	3.8	6.2654	3.8181	3.6213	2.3676	2.4972	2.6335	2

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Date          2
USD_MYR       2
GBP_MYR       2
EUR_MYR       2
JPY100_MYR    2
CHF_MYR       2
AUD_MYR       2
CAD_MYR       2
SGD_MYR       2
HKD100_MYR    2
dtype: int64
```

In [8]:
```
df = df.dropna().reset_index(drop=True)
df.isna().sum()
```

Out[8]:
```
Date          0
USD_MYR       0
GBP_MYR       0
EUR_MYR       0
JPY100_MYR    0
CHF_MYR       0
AUD_MYR       0
CAD_MYR       0
SGD_MYR       0
HKD100_MYR    0
dtype: int64
```

In [9]:
```
df.dtypes
```

Out[9]:
```
Date          object
USD_MYR       object
GBP_MYR       object
EUR_MYR       object
JPY100_MYR    object
CHF_MYR       object
AUD_MYR       object
CAD_MYR       object
SGD_MYR       object
HKD100_MYR    object
dtype: object
```

In [10]:
```
#B. Data Visualisation
```

In [11]:
```
for col in df.columns[1:]:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [12]:
```
df['Date'] = pd.to_datetime(df['Date'])
df['month'] = df['Date'].dt.month
df['year'] = df['Date'].dt.year
df['month_year'] = df['Date'].dt.to_period('M')
```

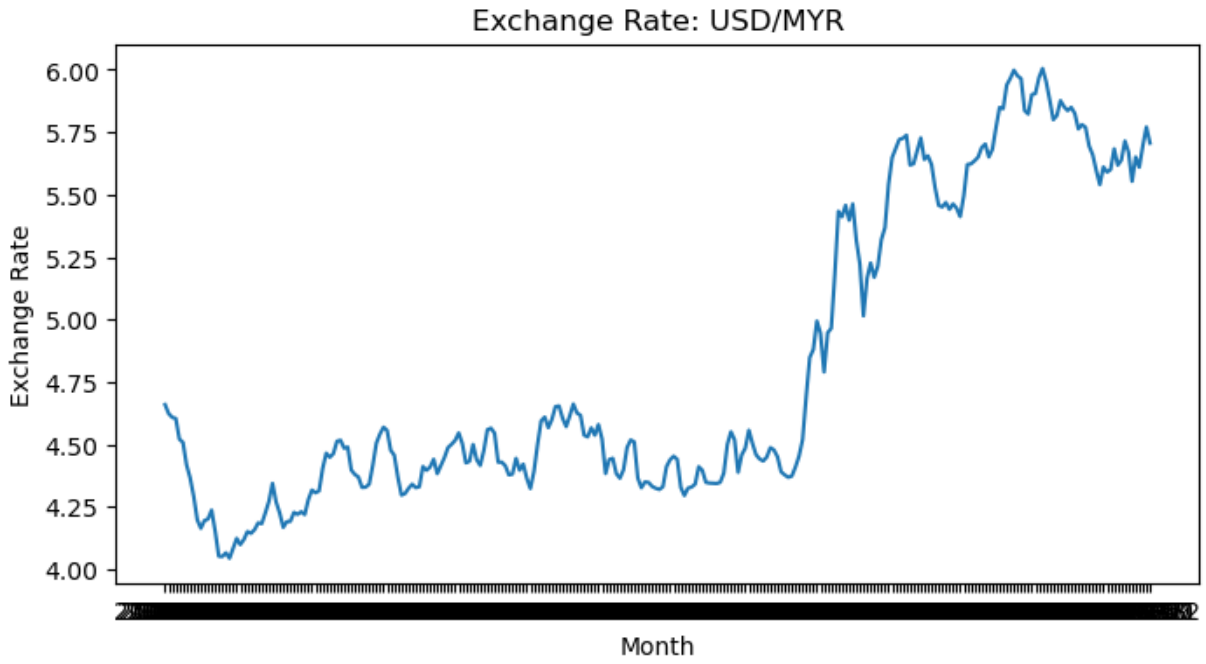
In [13]:
```
df_groupby_myr = df.groupby('month_year').USD_MYR.mean().reset_index()
df_groupby_myr
```

Out[13]:

	month_year	USD_MYR
0	2000-01	4.659383
1	2000-02	4.623653
2	2000-03	4.608377
3	2000-04	4.603860
4	2000-05	4.521210
...	...	...
271	2022-08	5.650715
272	2022-09	5.609749
273	2022-10	5.696011
274	2022-11	5.770798
275	2022-12	5.706441

276 rows × 2 columns

In [14]:
```
from matplotlib import pyplot as plt

x = df_groupby_myr['month_year'].astype(str)
y = df_groupby_myr['USD_MYR']

plt.figure(figsize=(8,4))
plt.plot(x, y)
plt.title("Exchange Rate: USD/MYR")
plt.xlabel("Month")
plt.ylabel("Exchange Rate")
plt.show()
```



In [15]:
```
df.columns
```

Out[15]:
```
Index(['Date', 'USD_MYR', 'GBP_MYR', 'EUR_MYR', 'JPY100_MYR', 'CHF_MYR',
       'AUD_MYR', 'CAD_MYR', 'SGD_MYR', 'HKD100_MYR', 'month', 'year',
       'month_year'],
      dtype='object')
```

In [16]:

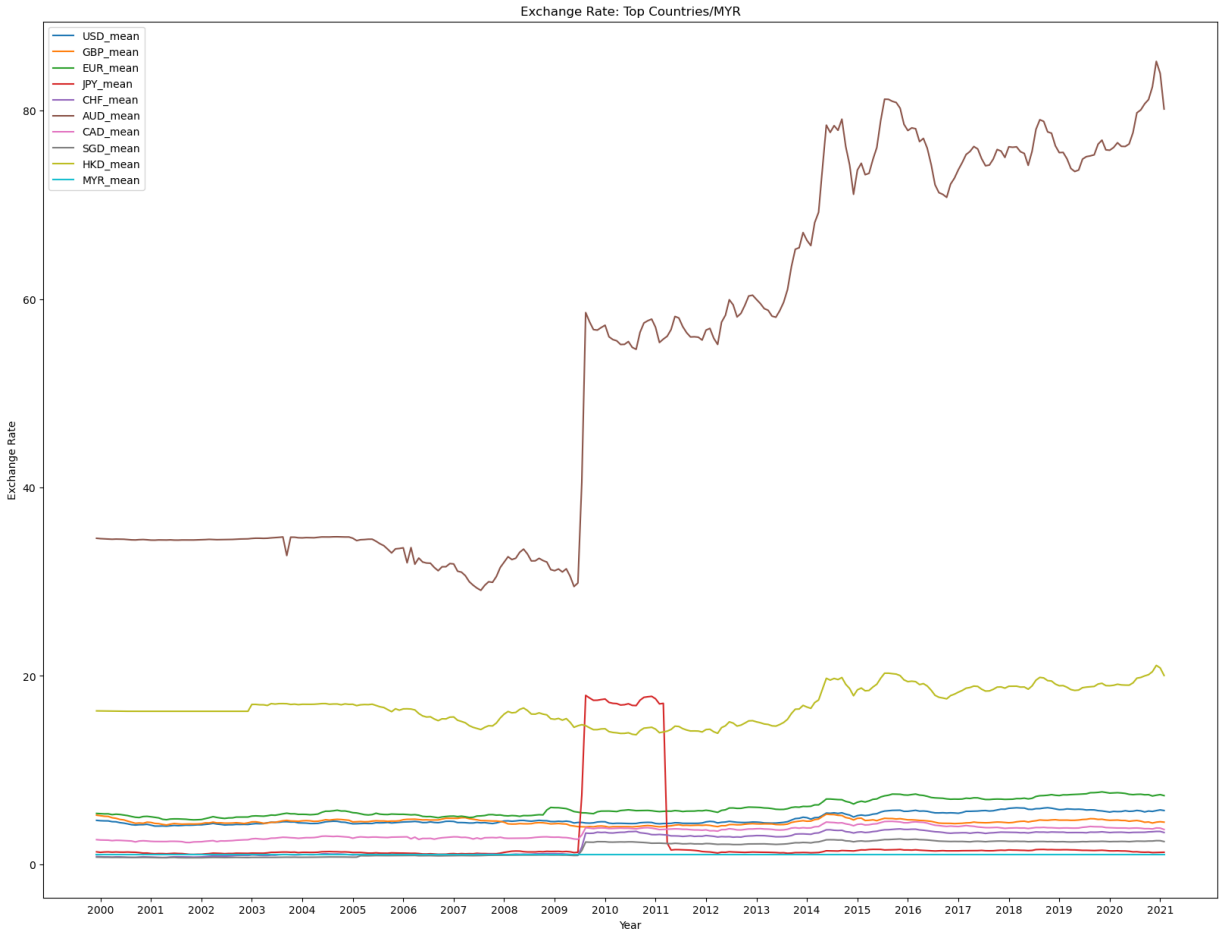
```
groupby_all = df.groupby('month_year').apply(lambda x: pd.Series({
                                                    'USD_mean': x['USD_MYR']
                                                    'GBP_mean': x['GBP_MYR']
                                                    'EUR_mean': x['EUR_MYR']
                                                    'JPY_mean': x['JPY100_MY
                                                    'CHF_mean': x['CHF_MYR']
                                                    'AUD_mean': x['AUD_MYR']
                                                    'CAD_mean': x['CAD_MYR']
                                                    'SGD_mean': x['SGD_MYR']
                                                    'HKD_mean': x['HKD100_MY
})).reset_index()

groupby_all = pd.DataFrame(groupby_all)

groupby_all['MYR_mean'] = [1 for i in range(276)]
groupby_all.head()
```

Out[16]:

	month_year	USD_mean	GBP_mean	EUR_mean	JPY_mean	CHF_mean	AUD_mean	CAD_mea
0	2000-01	4.659383	5.209283	5.394987	1.318802	0.815958	34.607715	2.61110
1	2000-02	4.623653	5.150284	5.372135	1.268332	0.793151	34.569567	2.57654
2	2000-03	4.608377	5.096289	5.349577	1.300361	0.779052	34.547047	2.56835
3	2000-04	4.603860	5.079511	5.354026	1.314547	0.778458	34.528688	2.55891
4	2000-05	4.521210	4.938006	5.269411	1.284524	0.753359	34.507037	2.50774

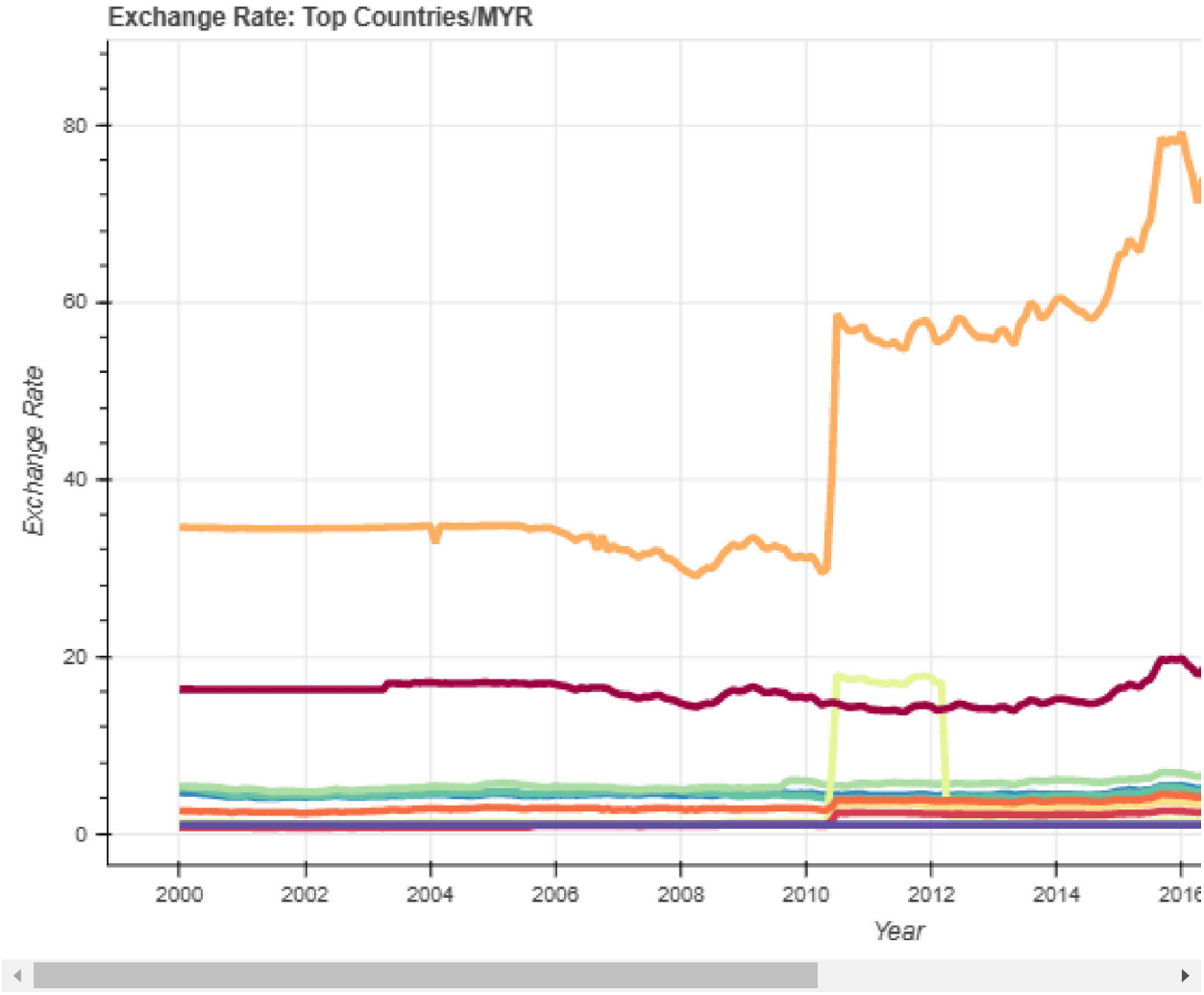
In [17]:

```
y_all = groupby_all[['USD_mean', 'GBP_mean', 'EUR_mean', 'JPY_mean', 'CHF_me
labels = ["USD_mean", "GBP_mean", "EUR_mean", "JPY_mean", "CHF_mean", "AUD_m
x_ticks = list(range(1, 280, 13))
x_ticklabels = [x for x in range(2000, 2022)]

plt.figure(figsize=(20,15))
ax = plt.subplot()

plt.plot(x, y_all)
ax.set_xticks(x_ticks)
ax.set_xticklabels(x_ticklabels)
plt.legend(labels)
plt.title("Exchange Rate: Top Countries/MYR")
plt.xlabel("Year")
plt.ylabel("Exchange Rate")

plt.show()
```



In [18]:

```
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.models.annotations import Title
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.palettes import Spectral10

from datetime import datetime

groupby_all['month_year'] = groupby_all['month_year'].astype(str)
groupby_all['month_year'] = pd.to_datetime(groupby_all['month_year'])

source = ColumnDataSource(groupby_all)

output_notebook()

p = figure(plot_height = 500, \
           plot_width = 900, \
           x_axis_type = 'datetime')

p.line(x='month_year', y='USD_mean', legend_label = 'USD_mean', source=sourc
p.line(x='month_year', y='GBP_mean', legend_label = 'GBP_mean', source=sourc
p.line(x='month_year', y='EUR_mean', legend_label = 'EUR_mean', source=sourc
p.line(x='month_year', y='JPY_mean', legend_label = 'JPY_mean', source=sourc
p.line(x='month_year', y='CHF_mean', legend_label = 'CHF_mean', source=sourc
p.line(x='month_year', y='AUD_mean', legend_label = 'AUD_mean', source=sourc
p.line(x='month_year', y='CAD_mean', legend_label = 'CAD_mean', source=sourc
p.line(x='month_year', y='SGD_mean', legend_label = 'SGD_mean', source=sourc
p.line(x='month_year', y='HKD_mean', legend_label = 'HKD_mean', source=sourc
p.line(x='month_year', y='MYR_mean', legend_label = 'MYR_mean', source=sourc


p.xaxis[0].ticker.desired_num_ticks = 10

p.xaxis.axis_label = 'Year'
p.yaxis.axis_label = 'Exchange Rate'

t = Title()
t.text = 'Exchange Rate: Top Countries/MYR'
p.title = t

hover = HoverTool(tooltips = [
                  ('USD_mean', '@USD_mean'),\
                  ('GBP_mean', '@GBP_mean'),\
                  ('EUR_mean', '@EUR_mean'),\
                  ('JPY_mean', '@JPY_mean'),\
                  ('CHF_mean', '@CHF_mean'),\
                  ('AUD_mean', '@AUD_mean'),\
                  ('CAD_mean', '@CAD_mean'),\
                  ('SGD_mean', '@SGD_mean'),\
                  ('HKD_mean', '@HKD_mean'),\
                  ('MYR_mean', '@MYR_mean')])
p.add_tools(hover)
show(p)
```



In [19]:
```
1 #C. Hypothesis Testing
```

In [20]:
```
1 usd_aud = df.loc[:,['USD_MYR','AUD_MYR']]
2 print(usd_aud)
```

```
       USD_MYR   AUD_MYR
0       3.8000    2.5002
1       3.8000    2.5021
2       3.8000    2.4915
3       3.8000    2.5002
4       3.8000    2.4854
...        ...       ...
17008   0.0187  120.5690
17009   0.0188  120.3763
17010   0.0188  120.6643
17011   0.0187  120.3169
17012   0.0187  120.1601

[17013 rows x 2 columns]
```

In [21]:
```
usd = np.array(usd_aud.iloc[:,0])
aud = np.array(usd_aud.iloc[:,1])

import statsmodels.stats.power as sms
from statsmodels import stats

#find sample size through power analysis
n = sms.TTestPower().solve_power(0.3, power=0.9, alpha=0.05)
print(n)
```

```
118.68650942951783
```

In [22]:
```
#generate sample sets from population
aud_sample = np.random.choice(aud, size=118)
usd_sample = np.random.choice(usd, size=118)

print(np.array(list(zip(aud_sample, usd_sample))))
```

```
[[9.039160e+01 3.140000e+00]
 [9.084940e+01 1.011290e+01]
 [8.684750e+01 4.475000e+00]
 [1.196355e+02 3.800000e+00]
 [0.000000e+00 1.037790e+01]
 [9.371540e+01 2.160000e-02]
 [0.000000e+00 3.800000e+00]
 [2.157900e+00 3.779200e+00]
 [2.651100e+00 3.800000e+00]
 [8.166010e+01 4.090500e+00]
 [1.069396e+02 3.028100e+00]
 [8.607240e+01 1.034320e+01]
 [3.004800e+00 2.140000e-02]
 [9.359580e+01 2.110000e-02]
 [2.928200e+00 1.810000e-02]
 [8.249720e+01 0.000000e+00]
 [2.819800e+00 9.991400e+00]
 [2.961600e+00 1.810000e-02]
 [2.888800e+00 1.009160e+01]
```

In [23]:
```
from scipy.stats import shapiro, pearsonr

#test normality of each sample set (H0: normally distributed)
stat, p1 = shapiro(aud_sample)
stat, p2 = shapiro(usd_sample)
print(p1)
print(p2)

#test whether the sample sets are correlated or not (H0: not correlated)
stat, p3 = pearsonr(aud_sample, usd_sample)
print(p3)
```

```
7.05752622141842e-13
2.89434587408266e-09
0.08451891450627709
```

In [24]:
```
#scipy
from scipy.stats import ttest_ind
tstat, pval = ttest_ind(aud_sample, usd_sample)
print('%.10f' %pval)

#statsmodels
import statsmodels.stats.api as sm
tstat, pval2, df = sm.ttest_ind(aud_sample, usd_sample)
print('%.30f' %pval2)
```

```
0.0000000000
0.000000000000000000091361534238
```

In [25]:
```
#D. Machine Learning Models
```

In [26]:
```
ir_df = pd.read_csv("interbank-intraday-rates-2018-2022.csv")
ir_df
```

Out[26]:

	Trading_Date	Highest_Rate	Lowest_Rate
0	2-Jan-18	4.045	4.0175
1	3-Jan-18	4.028	4.0140
2	4-Jan-18	4.028	4.0060
3	5-Jan-18	4.005	3.9900
4	8-Jan-18	3.999	3.9860
...	...	...	...
1189	21-Dec-22	4.441	4.4350
1190	22-Dec-22	4.435	4.4220
1191	23-Dec-22	4.435	4.4200
1192	28-Dec-22	4.440	4.4180
1193	29-Dec-22	4.429	4.4200

1194 rows × 3 columns

In [27]:
```
ir_df.dtypes
```

Out[27]:
```
Trading_Date     object
Highest_Rate    float64
Lowest_Rate     float64
dtype: object
```

In [28]:
```
for col in ir_df.columns[1:]:
    ir_df[col] = pd.to_numeric(ir_df[col], errors='coerce')
```

In [29]:
```
ir_df['Trading_Date'] = pd.to_datetime(ir_df['Trading_Date'])
ir_df['month'] = ir_df['Trading_Date'].dt.month
ir_df['year'] = ir_df['Trading_Date'].dt.year
ir_df['month_year'] = ir_df['Trading_Date'].dt.to_period('M')
```

In [30]:
```
1 ir_df
```

Out[30]:

	Trading_Date	Highest_Rate	Lowest_Rate	month	year	month_year
0	2018-01-02	4.045	4.0175	1	2018	2018-01
1	2018-01-03	4.028	4.0140	1	2018	2018-01
2	2018-01-04	4.028	4.0060	1	2018	2018-01
3	2018-01-05	4.005	3.9900	1	2018	2018-01
4	2018-01-08	3.999	3.9860	1	2018	2018-01
...	...	...	...	...	...	...
1189	2022-12-21	4.441	4.4350	12	2022	2022-12
1190	2022-12-22	4.435	4.4220	12	2022	2022-12
1191	2022-12-23	4.435	4.4200	12	2022	2022-12
1192	2022-12-28	4.440	4.4180	12	2022	2022-12
1193	2022-12-29	4.429	4.4200	12	2022	2022-12

In [31]:
```
1 df_groupby_ir = ir_df.groupby('month_year').Highest_Rate.mean().reset_index(
2 df_groupby_ir.head()
```

Out[31]:

	month_year	Highest_Rate
0	2018-01	3.966976
1	2018-02	3.918233
2	2018-03	3.908182
3	2018-04	3.889775
4	2018-05	3.969250

In [32]:
```
1 x_ir = df_groupby_ir['Highest_Rate']
2 x_ir = np.array(x_ir).reshape(-1,1)
```

In [33]:
```
1 usd_myr_fx = df_groupby_myr[(df_groupby_myr['month_year'] >= '2018-01') & (d
2 usd_myr = usd_myr_fx['USD_MYR']
3
4 y_fx = usd_myr
```

In [34]:
```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
```

In [35]:
```
1 x_train, x_test, y_train, y_test = train_test_split(x_ir, y_fx, train_size=0
```

Model to be used in this dataet

1. Linear Regression
2. Polynomial Regression

3. Ridge Regression & Lasso Regression
4. K-Neasrest Neighbour Regression
5. Support Vector Regression

In [36]:
```
R_squared=[]
MAE=[]
MSE=[]
RMSE=[]
Model = []
```

Linear Regression

In [37]:
```
model = LinearRegression()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
print(y_predict)
```

```
[5.70250591 5.705314   5.64749862 5.7190307  5.722121   5.71360114
 5.70990829 5.68609478 5.70389803 5.76175054 5.69188168 5.77177564]
```

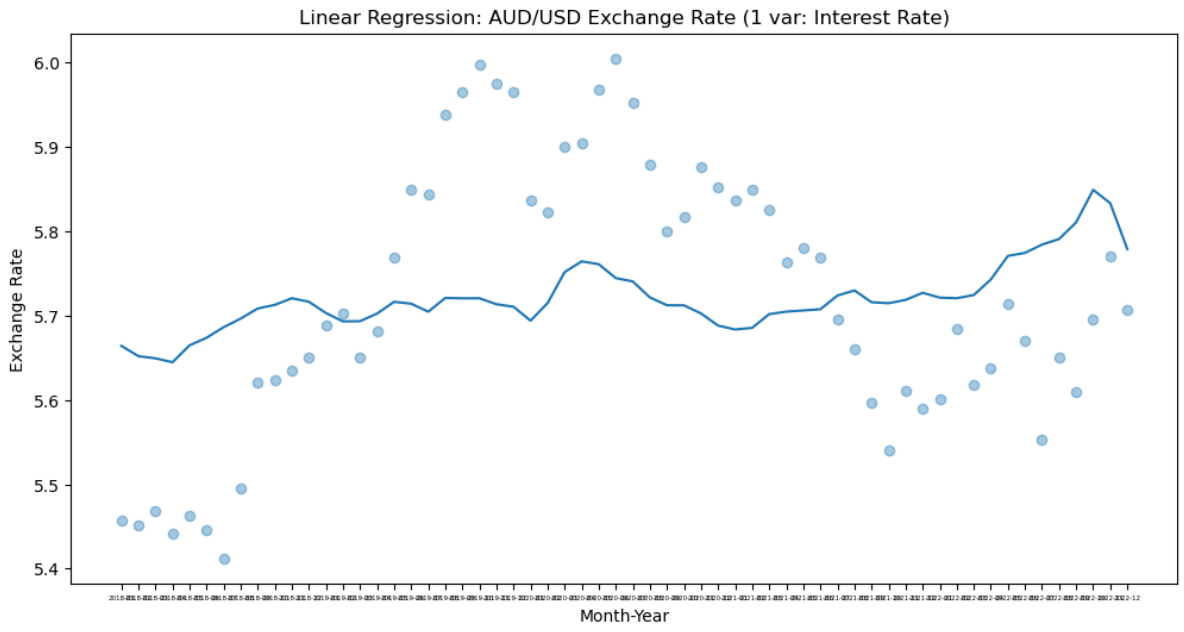
In [38]:
```
print("R-squared of training data is: " + str(model.score(x_train, y_train)))
print("R-squared of testing data is: " + str(model.score(x_test, y_test)))
```

```
R-squared of training data is: 0.06094924772473809
R-squared of testing data is: 0.06983088859191167
```

In [39]:
```
model = LinearRegression()
model.fit(x_ir, y_fx)
y_fx_predict = model.predict(x_ir)

print("R-squared of the entire dataset is: " + str(model.score(x_ir, y_fx)))
```

```
R-squared of the entire dataset is: 0.06231308561643811
```

In [40]:
```
month_year = usd_myr_fx['month_year']
month_year = month_year.astype(str)

plt.figure(figsize=(12,6))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_fx_predict)
plt.title("Linear Regression: AUD/USD Exchange Rate (1 var: Interest Rate)")
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=4)
plt.show()
```



In [41]:
```
import sklearn.metrics as metrics

mae_linear = metrics.mean_absolute_error(y_fx, y_fx_predict)
mse_linear = metrics.mean_squared_error(y_fx, y_fx_predict)
rmse_linear = np.sqrt(mse_linear) # or mse**(0.5)
r2_linear = metrics.r2_score(y_fx, y_fx_predict)

print("Results of sklearn.metrics: Linear")
print("MAE:",mae_linear)
print("MSE:", mse_linear)
print("RMSE:", rmse_linear)
print("R-Squared:", r2_linear)
```

```
Results of sklearn.metrics: Linear
MAE: 0.13781033888268673
MSE: 0.024034915726698233
RMSE: 0.15503198291545597
R-Squared: 0.06231308561643811
```

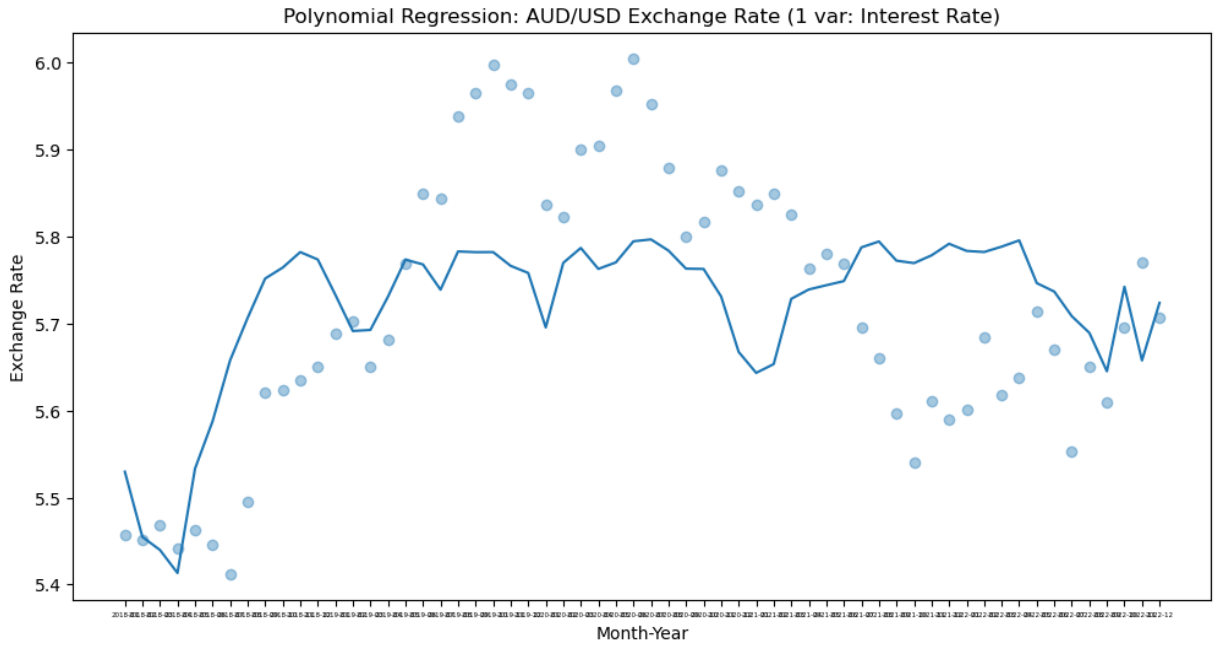
In [42]:
```
MAE.append(mae_linear)
MSE.append(mse_linear)
RMSE.append(rmse_linear)
R_squared.append(r2_linear)
Model.append('Linear')
```

Polynomial Regression

In [43]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=4)
x_poly = poly.fit_transform(x_ir)
model_poly = LinearRegression()
model_poly.fit(x_poly, y_fx)
y_pred = model_poly.predict(x_poly)
print(model_poly.score(x_poly, y_fx))
```

```
0.30482494556026196
```

In [44]:
```
month_year = usd_myr_fx['month_year']
month_year = month_year.astype(str)

plt.figure(figsize=(12,6))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_pred)
plt.title("Polynomial Regression: AUD/USD Exchange Rate (1 var: Interest Rat
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=4)
plt.show()
```



In [45]:

```python
import sklearn.metrics as metrics

mae_poly = metrics.mean_absolute_error(y_fx, y_pred)
mse_poly = metrics.mean_squared_error(y_fx, y_pred)
rmse_poly = np.sqrt(mse_poly) # or mse**(0.5)
r2_poly = metrics.r2_score(y_fx, y_pred)

print("Results of sklearn.metrics: Polynomial")
print("MAE:",mae_poly)
print("MSE:", mse_poly)
print("RMSE:", rmse_poly)
print("R-Squared:", r2_poly)
```

```
Results of sklearn.metrics: Polynomial
MAE: 0.11444499943671242
MSE: 0.0178188194721115
RMSE: 0.13348715096259828
R-Squared: 0.30482494556026196
```

In [46]:

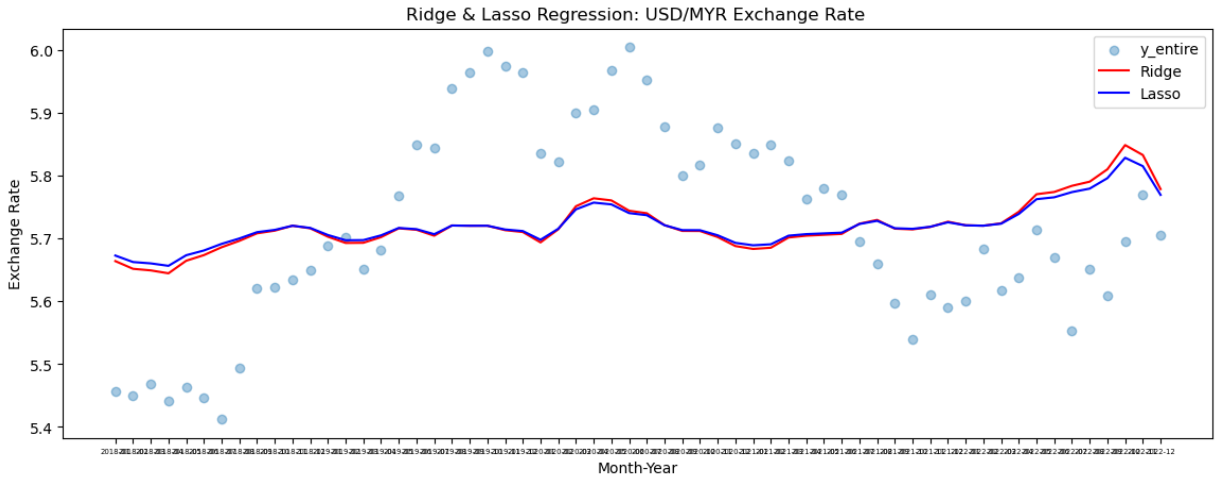
```python
MAE.append(mae_poly)
MSE.append(mse_poly)
RMSE.append(rmse_poly)
R_squared.append(r2_poly)
Model.append('Polynomial')
```

Ridge Regression & Lasso Regression

In [47]:

```
from sklearn.linear_model import Ridge, Lasso

ridge = Ridge(alpha=0.001)
ridge.fit(x_ir, y_fx)

y_fx_ridge = ridge.predict(x_ir)
print(ridge.score(x_ir, y_fx))

lasso = Lasso(alpha=0.001)
lasso.fit(x_ir, y_fx)

y_fx_lasso = lasso.predict(x_ir)
print(lasso.score(x_ir, y_fx))

labels = ["y_entire","Ridge","Lasso"]

plt.figure(figsize=(14,5))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_fx_ridge, color='r')
plt.plot(month_year, y_fx_lasso, color='b')
plt.legend(labels)
plt.title("Ridge & Lasso Regression: USD/MYR Exchange Rate ")
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=5)
plt.show()
```

```
0.062313058289344525
0.060761908867496994
```



In [48]:
```
import sklearn.metrics as metrics

mae_ridge = metrics.mean_absolute_error(y_fx, y_fx_ridge)
mse_ridge = metrics.mean_squared_error(y_fx, y_fx_ridge)
rmse_ridge = np.sqrt(mse_ridge) # or mse**(0.5)
r2_ridge = metrics.r2_score(y_fx, y_fx_ridge)

print("Results of sklearn.metrics: Ridge")
print("MAE:",mae_ridge)
print("MSE:", mse_ridge)
print("RMSE:", rmse_ridge)
print("R-Squared:", r2_ridge)
```

```
Results of sklearn.metrics: Ridge
MAE: 0.13780718221029928
MSE: 0.024034916427149933
RMSE: 0.15503198517451144
R-Squared: 0.062313058289344525
```

In [49]:
```
MAE.append(mae_ridge)
MSE.append(mse_ridge)
RMSE.append(rmse_ridge)
R_squared.append(r2_ridge)
Model.append('Ridge')
```

In [50]:
```
import sklearn.metrics as metrics

mae_lasso = metrics.mean_absolute_error(y_fx, y_fx_lasso)
mse_lasso = metrics.mean_squared_error(y_fx, y_fx_lasso)
rmse_lasso = np.sqrt(mse_lasso) # or mse**(0.5)
r2_lasso = metrics.r2_score(y_fx, y_fx_lasso)

print("Results of sklearn.metrics: Lasso")
print("MAE:",mae_lasso)
print("MSE:", mse_lasso)
print("RMSE:", rmse_lasso)
print("R-Squared:", r2_lasso)
```

```
Results of sklearn.metrics: Lasso
MAE: 0.13705825992359108
MSE: 0.024074675695474723
RMSE: 0.15516016143158243
R-Squared: 0.060761908867496994
```

In [51]:
```
MAE.append(mae_lasso)
MSE.append(mse_lasso)
RMSE.append(rmse_lasso)
R_squared.append(r2_lasso)
Model.append('Lasso')
```

K-Nearest Neighbour Regression

In [52]:
```
from sklearn.neighbors import KNeighborsRegressor
knn_model = KNeighborsRegressor(n_neighbors=5,metric='euclidean')
knn_model.fit(x_train, y_train)
knn_pred = knn_model.predict(x_test)
```

In [53]:
```
print("R-squared of training data is: " + str(knn_model.score(x_train, y_tra
print("R-squared of testing data is: " + str(knn_model.score(x_test, y_test)
```

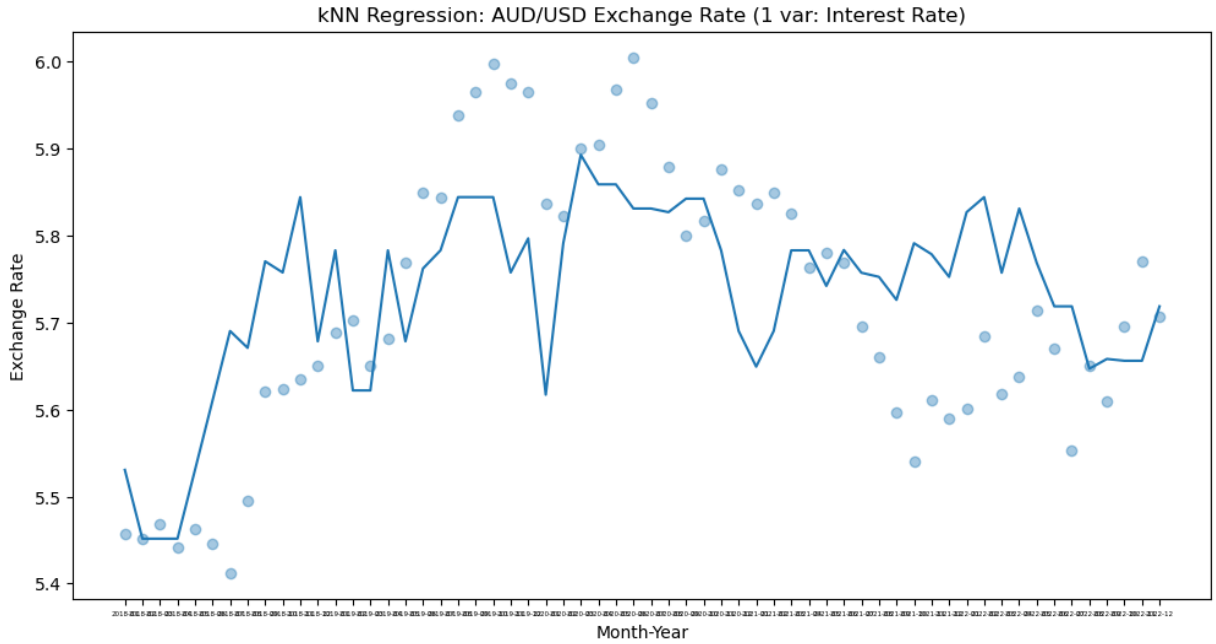
```
R-squared of training data is: 0.426644008702908
R-squared of testing data is: 0.10020811407667907
```

In [54]:
```
model.fit(x_ir, y_fx)
y_knn = knn_model.predict(x_ir)
```

In [55]:
```
print("R-squared of the entire dataset is: " + str(knn_model.score(x_ir, y_f
```

```
R-squared of the entire dataset is: 0.38673773790801846
```

In [56]:
```
month_year = usd_myr_fx['month_year']
month_year = month_year.astype(str)

plt.figure(figsize=(12,6))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_knn)
plt.title("kNN Regression: AUD/USD Exchange Rate (1 var: Interest Rate)")
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=4)
plt.show()
```



In [57]:
```
import sklearn.metrics as metrics

mae_knn = metrics.mean_absolute_error(y_fx, y_knn)
mse_knn = metrics.mean_squared_error(y_fx, y_knn)
rmse_knn = np.sqrt(mse_knn) # or mse**(0.5)
r2_knn = metrics.r2_score(y_fx, y_knn)

print("Results of sklearn.metrics: kNN")
print("MAE:",mae_knn)
print("MSE:", mse_knn)
print("RMSE:", rmse_knn)
print("R-Squared:", r2_knn)
```

```
Results of sklearn.metrics: kNN
MAE: 0.10378981723507695
MSE: 0.015719219882080818
RMSE: 0.12537631308218
R-Squared: 0.38673773790801846
```

In [58]:
```
MAE.append(mae_knn)
MSE.append(mse_knn)
RMSE.append(rmse_knn)
R_squared.append(r2_knn)
Model.append('KNN')
```

Support Vector Regeression

In [59]:
```
from sklearn.svm import SVR
svr_model = SVR(kernel = 'rbf')
svr_model.fit(x_train, y_train)
svr_pred = svr_model.predict(x_test)
print(svr_pred)
```

```
[5.73497741 5.73975219 5.44023252 5.76442358 5.77444343 5.75203555
 5.7462519  5.6650441  5.73748163 5.84743363 5.69983773 5.75582786]
```

In [60]:
```
print("R-squared of training data is: " + str(svr_model.score(x_train, y_tra
print("R-squared of testing data is: " + str(svr_model.score(x_test, y_test)
```

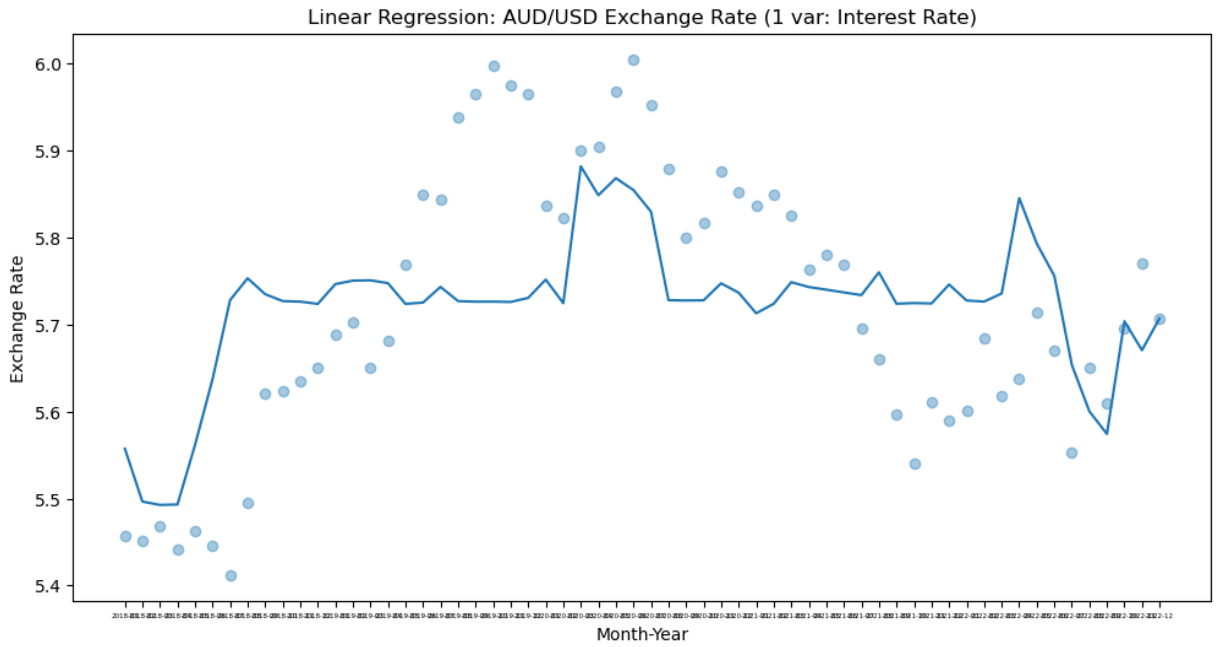
```
R-squared of training data is: 0.39299413096649005
R-squared of testing data is: 0.2076951343642901
```

In [61]:
```
svr_model = SVR(kernel = 'rbf')
svr_model.fit(x_ir, y_fx)
y_svr = svr_model.predict(x_ir)

print("R-squared of the entire dataset is: " + str(svr_model.score(x_ir, y_f
```

```
R-squared of the entire dataset is: 0.355006238035672
```

In [62]:

```
month_year = usd_myr_fx['month_year']
month_year = month_year.astype(str)

plt.figure(figsize=(12,6))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_svr)
plt.title("Linear Regression: AUD/USD Exchange Rate (1 var: Interest Rate)")
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=4)
plt.show()
```



In [63]:

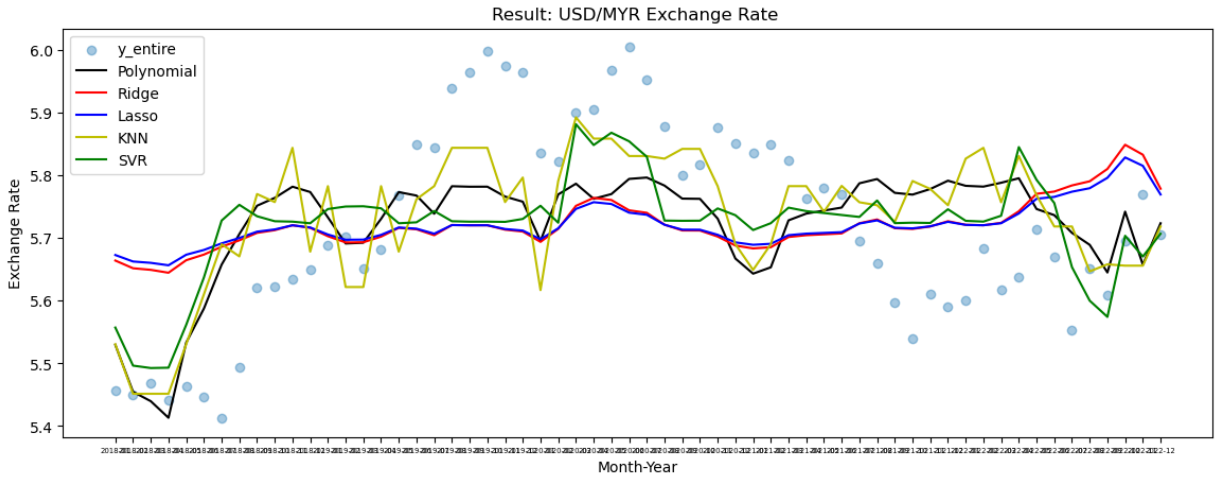
```
import sklearn.metrics as metrics

mae_svr = metrics.mean_absolute_error(y_fx, y_svr)
mse_svr = metrics.mean_squared_error(y_fx, y_svr)
rmse_svr = np.sqrt(mse_svr) # or mse**(0.5)
r2_svr = metrics.r2_score(y_fx, y_svr)

print("Results of sklearn.metrics: SVR")
print("MAE:",mae_svr)
print("MSE:", mse_svr)
print("RMSE:", rmse_svr)
print("R-Squared:", r2_svr)
```

```
Results of sklearn.metrics: SVR
MAE: 0.10831126565444975
MSE: 0.016532565907939525
RMSE: 0.1285790259254577
R-Squared: 0.355006238035672
```

In [64]:
```
MAE.append(mae_svr)
MSE.append(mse_svr)
RMSE.append(rmse_svr)
R_squared.append(r2_svr)
Model.append('SVR')
```

Result

In [65]:
```
labels = ["y_entire","Polynomial","Ridge","Lasso","KNN","SVR"]

plt.figure(figsize=(14,5))
plt.scatter(month_year, y_fx, alpha=0.4)
plt.plot(month_year, y_pred, color='black')
plt.plot(month_year, y_fx_ridge, color='r')
plt.plot(month_year, y_fx_lasso, color='b')
plt.plot(month_year, y_knn, color='y')
plt.plot(month_year, y_svr, color='g')
plt.legend(labels)
plt.title("Result: USD/MYR Exchange Rate ")
plt.xlabel("Month-Year")
plt.ylabel("Exchange Rate")
plt.xticks(fontsize=5)
plt.show()
```



Evaluation

In [66]:
```
print("Model Used:-")
Model
```

```
Model Used:-
```

Out[66]:
```
['Linear', 'Polynomial', 'Ridge', 'Lasso', 'KNN', 'SVR']
```

In [67]:

```
output=pd.DataFrame({'Model':Model,
                     'MAE':[mae_linear, mae_poly,mae_ridge, mae_lasso, mae_kn
                     'MSE':[mse_linear, mse_poly ,mse_ridge, mse_lasso, mse_k
                      'RMSE': [rmse_linear, rmse_poly ,rmse_ridge, rmse_lasso
                      'R_squared':[r2_linear, r2_poly , r2_ridge, r2_lasso, r
                     })
output=output.set_index('Model')
output
```

Out[67]:

Model	MAE	MSE	RMSE	R_squared
Linear	0.137810	0.024035	0.155032	0.062313
Polynomial	0.114445	0.017819	0.133487	0.304825
Ridge	0.137807	0.024035	0.155032	0.062313
Lasso	0.137058	0.024075	0.155160	0.060762
KNN	0.103790	0.015719	0.125376	0.386738
SVR	0.108311	0.016533	0.128579	0.355006

Based on the result that we have run, we conclude that KNN is the best model among other model. This is because MAE, MSE, RMSE of KNN is the lowest, which is 0.103790, 0.015719, 0.125376, 0.386738 while the R-Squared is the highest which is 0.386738. In addition, the best evaluation is RMSE due to the time series forecasting and measure the average difference between the predicted and actual values.

Furthermore, MAE also another option to evaluate the exchange rate. Both evaluations give a good indication of how accurate the model in predicting exchange rates.(in applied predictive modelling,Max Kuhn, J.Kjell)