Importing the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.ensemble import ExtraTreesRegressor
```

In [2]:
```
# function to plot the graph
import plotly_express as px
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

def plot_data(df,title,y):
    '''function for plotting gold data'''
    plot = px.line(df,
                   x="Date",
                   y=[y],
                   hover_name="Date",
                   line_shape="linear",
                   title=title)
    return plot
```

In [3]:
```
# yfinance is used to fetch data
import yfinance as yf
```

Importing Gold Price Data

In [4]:
```
#df = yf.download('GLD','2008-01-01','2022-10-30')


#create data drame to read data set
df = pd.read_csv('gld_price_data.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

Data set columns

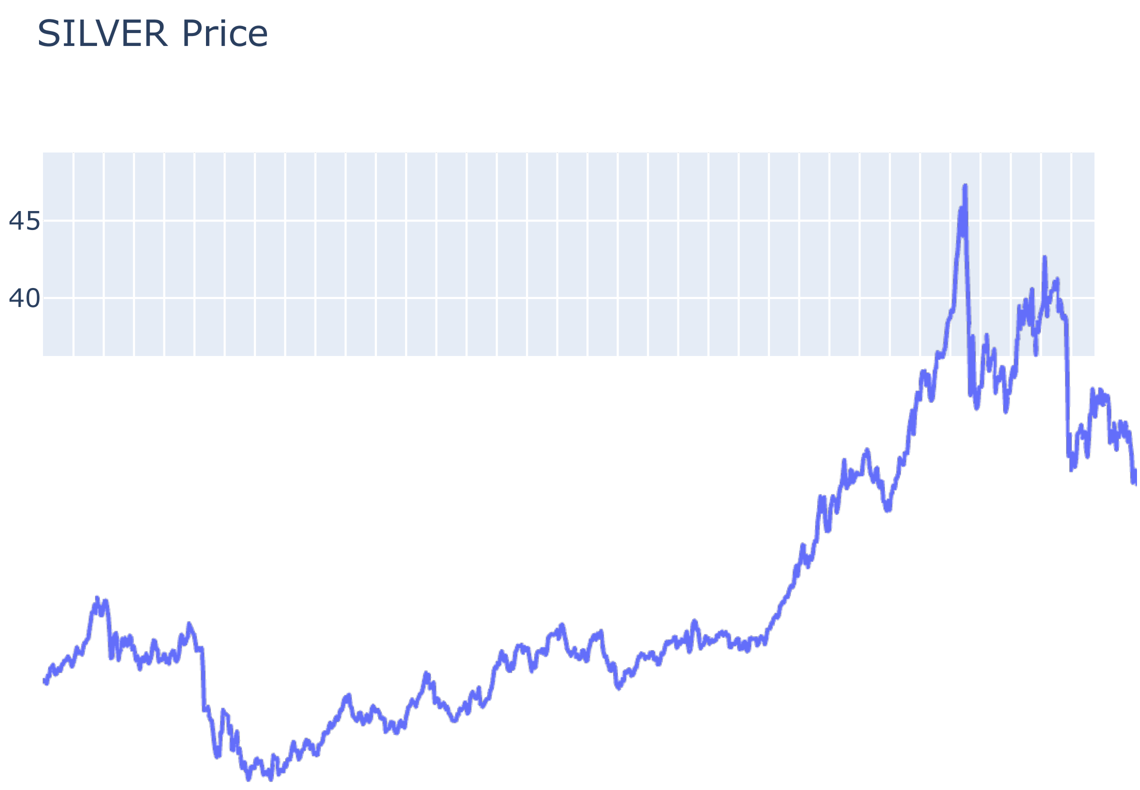
- Date - mm/dd/yyyy
- SPX - is a free-float weighted measurement stock market index of the 500 largest companies listed on stock exchanges in the United States.
- GLD - Gold Price
- USO - United States Oil Fund
- SLV - Silver Price
- EUR/USD - currency pair quotation of the Euro against the US

Data Exploration & Analysis

Plotting the raw data

In [6]:

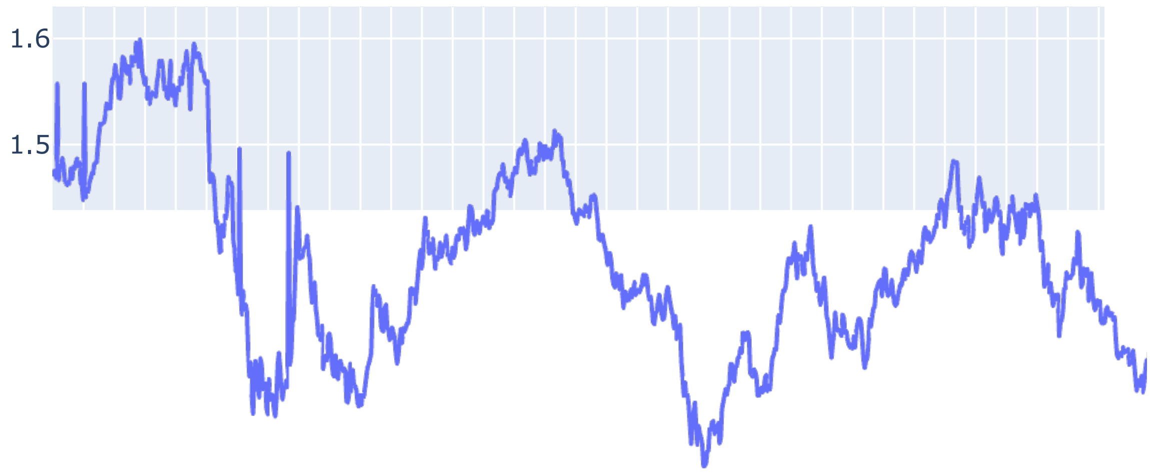
```
c1 = "SLV"
plot_data(df,'SILVER Price',c1)
```



Exploring the raw data

In [7]:
```
c5 = "EUR/USD"
plot_data(df,'Plot of EUR/USD Data',c5)
```

Plot of EUR/USD Data



In [8]:
```
# checking structure of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Date     2290 non-null   object 
 1   SPX      2290 non-null   float64
 2   GLD      2290 non-null   float64
 3   USO      2290 non-null   float64
 4   SLV      2290 non-null   float64
 5   EUR/USD  2290 non-null   float64
dtypes: float64(5), object(1)
memory usage: 107.5+ KB
```

In [9]:
```
# finding number of rows and column
df.shape
```

Out[9]: (2290, 6)

In [10]:
```
# describe df numerical columns
df.describe()
```

Out[10]:

	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000
mean	1654.315776	122.732875	31.842221	20.084997	1.283653
std	519.111540	23.283346	19.523517	7.092566	0.131547
min	676.530029	70.000000	7.960000	8.850000	1.039047
25%	1239.874969	109.725000	14.380000	15.570000	1.171313
50%	1551.434998	120.580002	33.869999	17.268500	1.303297
75%	2073.010070	132.840004	37.827501	22.882500	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

Finding unwanted collumns

In [11]:
```
df.head()
```

Out[11]:

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

We will not consider Date Feature and hence we will drop this feature during feature selection.

In [12]:
```
#finding the missing values
df.isna().sum()
```

Out[12]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

Result : We don't have any missing values.

Finding Features with only one value

In [13]:

```
for column in df.columns:
    print(column,df[column].nunique())
```

```
Date 2290
SPX 2277
GLD 1930
USO 1514
SLV 1331
EUR/USD 2066
```

Result : No feature found with only one value.

Finding Duplicated Data

In [14]:

```
df.duplicated().sum()
```

Out[14]:

```
0
```

Result : No duplicated data found.

In [15]:

```
# list of numerical variables
numerical_features = [feature for feature in df.columns if ((df[feature].d
print('Number of numerical variables: ', len(numerical_features))

# visualise the numerical variables
df[numerical_features].head()



discrete_feature=[feature for feature in numerical_features if len(df[featu
print("Discrete Variables Count: {}".format(len(discrete_feature)))

continuous_features=[feature for feature in numerical_features if feature n
print("Continuous feature Count {}".format(len(continuous_features)))
```

```
Number of numerical variables:  4
Discrete Variables Count: 0
Continuous feature Count 4
```

Results :

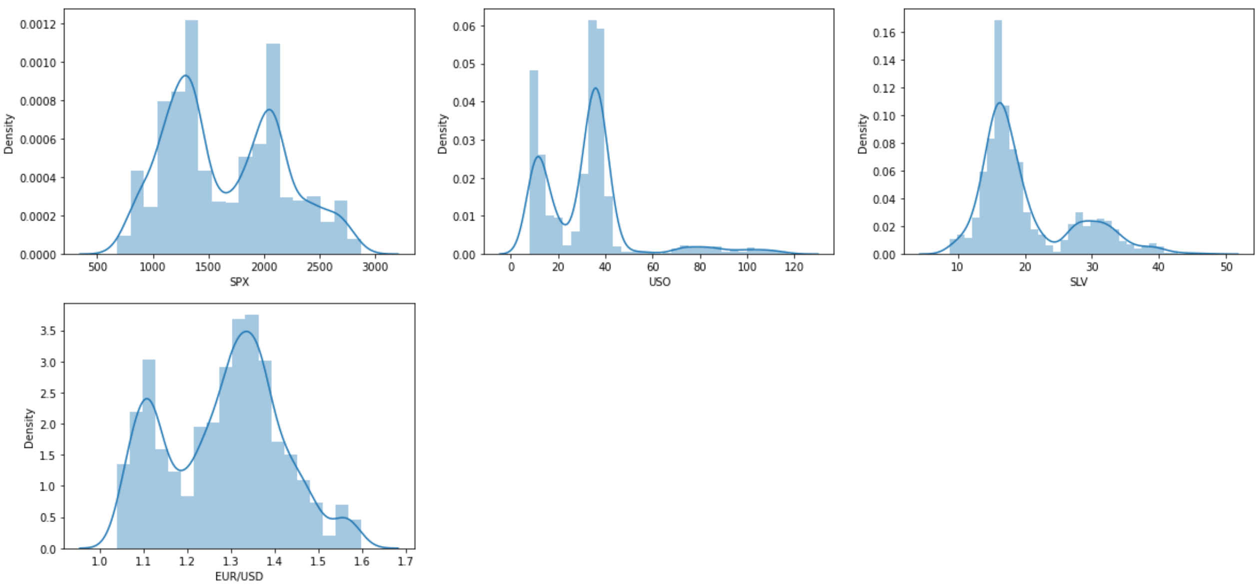
- There are 4 numerical features
- There are 0 Discrete Variables in give dataset
- There are 4 continuous numerical features

Distribution of Continuous Numerical Features

In [16]:

```
#plot a univariate distribution of continues observations
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for continuous_feature in continuous_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.distplot(df[continuous_feature])
    plt.xlabel(continuous_feature)
    plotnumber+=1
plt.show()
```

```
/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version
. Please adapt your code to use either `displot` (a figure-level function w
ith similar flexibility) or `histplot` (an axes-level function for histogra
ms).

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version
. Please adapt your code to use either `displot` (a figure-level function w
ith similar flexibility) or `histplot` (an axes-level function for histogra
ms).

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version
. Please adapt your code to use either `displot` (a figure-level function w
ith similar flexibility) or `histplot` (an axes-level function for histogra
ms).

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version
. Please adapt your code to use either `displot` (a figure-level function w
ith similar flexibility) or `histplot` (an axes-level function for histogra
ms).
```



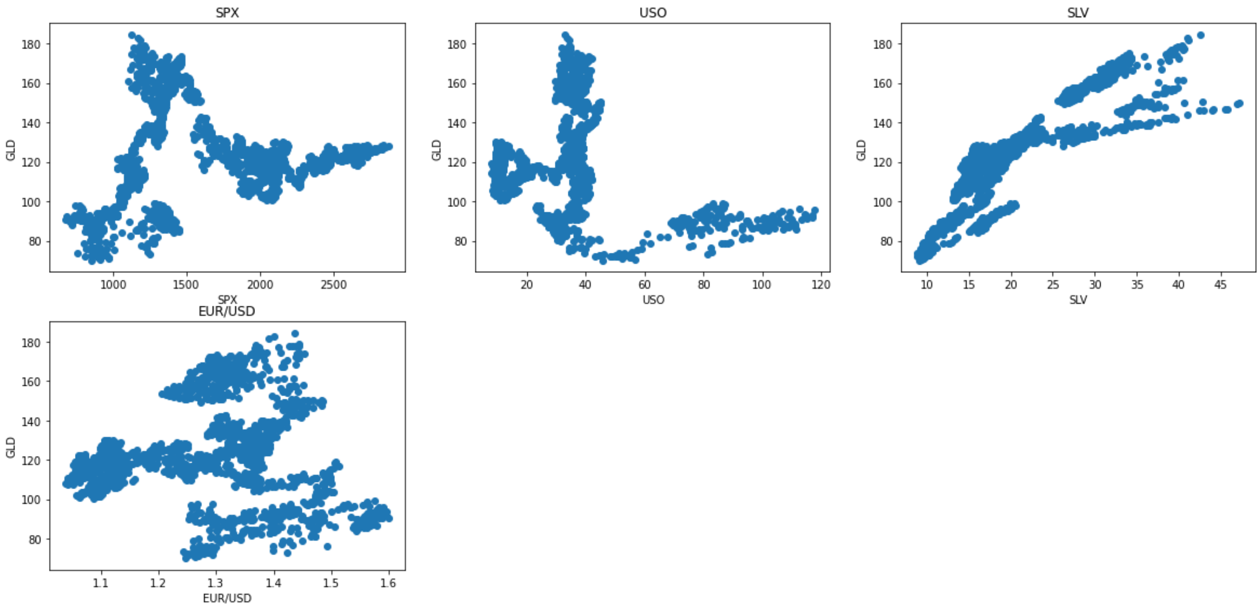
Results :

- USO heavily skewed towards right and seems to be have some outliers.
- It seems like the rest are distributed normally.

Relation between Continous numerical Features and Labels

In [17]:

```
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for feature in continuous_features:
    data=df.copy()
    ax = plt.subplot(12,3,plotnumber)
    plt.scatter(data[feature],data['GLD'])
    plt.xlabel(feature)
    plt.ylabel('GLD')
    plt.title(feature)
    plotnumber+=1
plt.show()
```



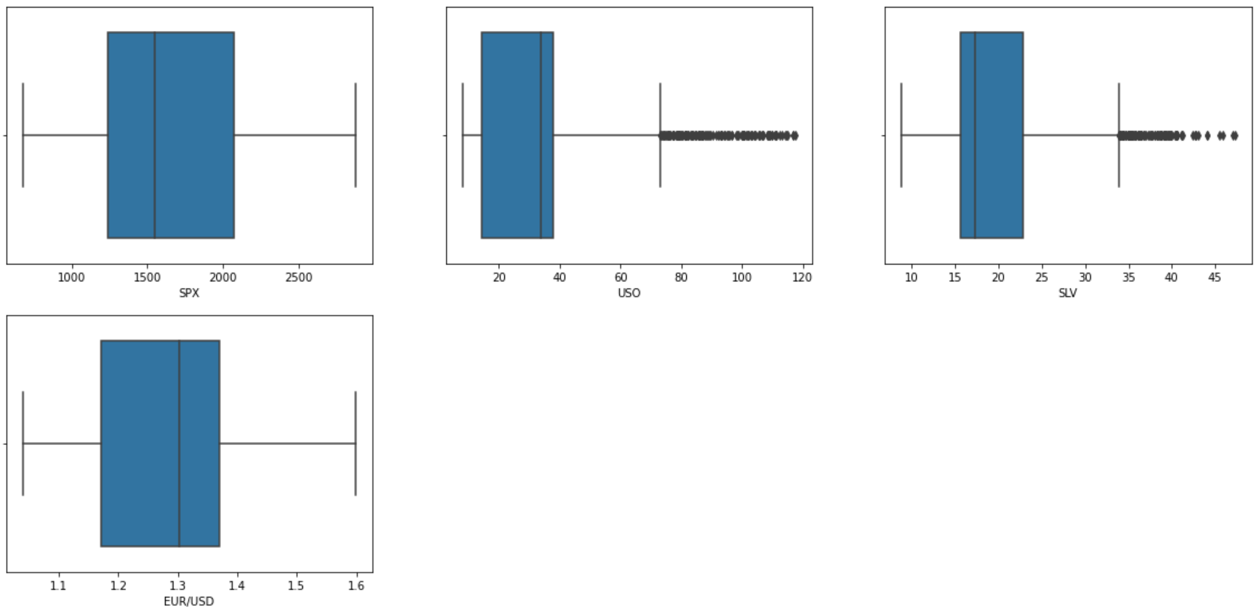
Result : It seems like SLV is passing linearly with Gold

Finding Data Outliners

In [18]:

```
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for numerical_feature in numerical_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.boxplot(df[numerical_feature])
    plt.xlabel(numerical_feature)
    plotnumber+=1
plt.show()
```

```
/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.

/Users/79_satya/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
```



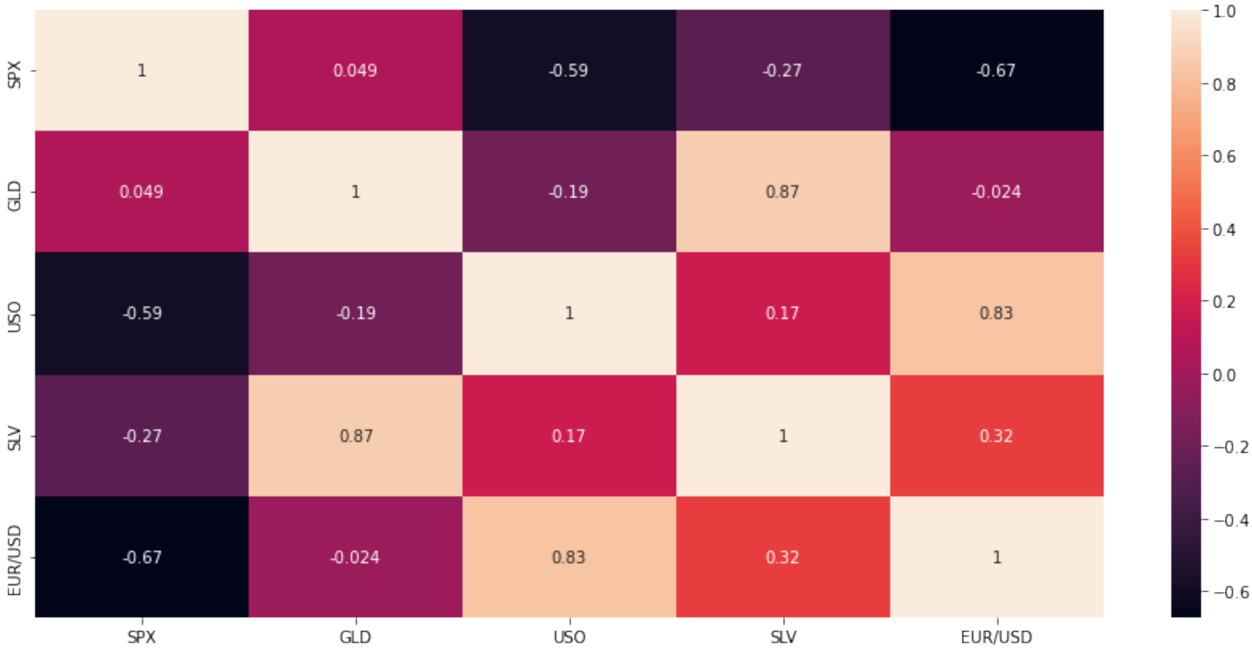
Results :

- It seems like USO and SLV have some outliners

Explore the Correlation between numerical features

In [19]:

```
## Checking for correlation
cor_mat=df.corr()
fig = plt.figure(figsize=(15,7))
sns.heatmap(cor_mat,annot=True)
plt.show()
```



In [20]:

```
print (cor_mat['GLD'].sort_values(ascending=False), '\n')
```

```
GLD        1.000000
SLV        0.866632
SPX        0.049345
EUR/USD   -0.024375
USO       -0.186360
Name: GLD, dtype: float64
```

Result : It seems SLV feature is heavily correlated with GLD

Data Preprocessing

In [21]:

```
data_preprocessed = df.copy()
```

In [22]:

```
data_preprocessed.isnull().mean() * 100
```

Out[22]:
```
Date       0.0
SPX        0.0
GLD        0.0
USO        0.0
SLV        0.0
EUR/USD    0.0
dtype: float64
```

Dropping the date collumn

In [23]:
```
data_preprocessed['Date'] = pd.to_datetime(data_preprocessed['Date'])
```

In [24]:
```
date_columns = ['Date']
num_columns = data_preprocessed.select_dtypes(include=['float64', 'int64']
target_col = 'GLD'
```

In [25]:
```
num_columns
```

Out[25]:
```
Index(['SPX', 'GLD', 'USO', 'SLV', 'EUR/USD'], dtype='object')
```

In [26]:
```
data_preprocessed.reset_index(drop=True, inplace=True)
```

In [27]:
```
data_preprocessed.drop(['Date'], axis=1, inplace=True)
```

Spliting data into Train and Test Sets

In [28]:
```
## train test split

X = data.drop(['Date','GLD'],axis=1)
Y = data['GLD']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2,
```

Visually inspecting the Train and test data

In [29]:
```
len(X_train)
```

Out[29]:
```
1832
```

In [30]:
```
len(X_test)
```

Out[30]: 458

In [31]:

```
X.head()
```

Out[31]:

	SPX	USO	SLV	EUR/USD
0	1447.160034	78.470001	15.180	1.471692
1	1447.160034	78.370003	15.285	1.474491
2	1411.630005	77.309998	15.167	1.475492
3	1416.180054	75.500000	15.053	1.468299
4	1390.189941	76.059998	15.590	1.557099

In [32]:

```
Y.head()
```

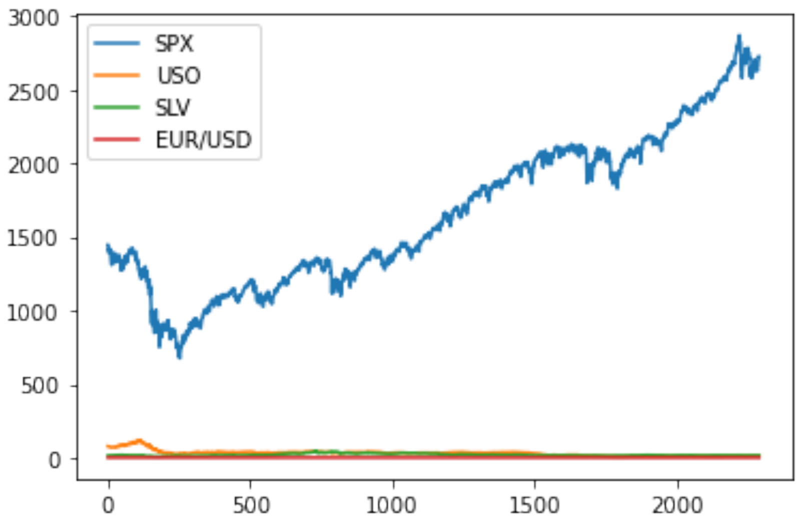
Out[32]:

```
0    84.860001
1    85.570000
2    85.129997
3    84.769997
4    86.779999
Name: GLD, dtype: float64
```

In [33]:

```
X.plot()
```

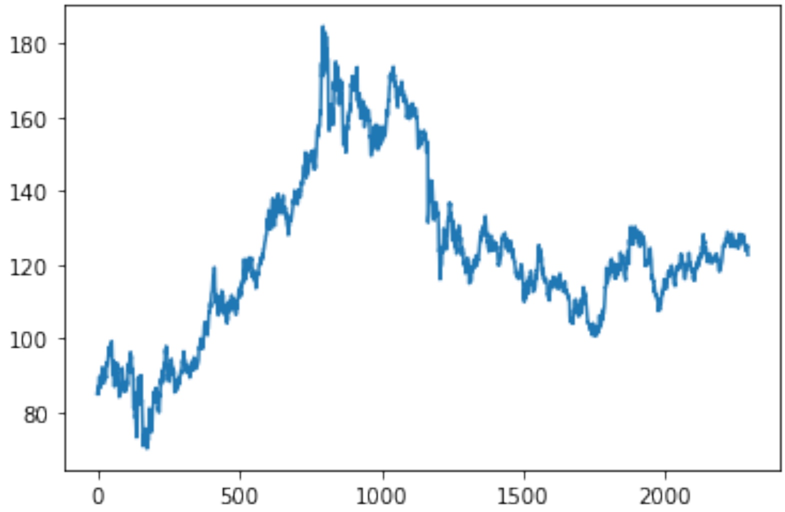
Out[33]: <AxesSubplot:>



In [34]:

```
Y.plot()
```

Out[34]: <AxesSubplot:>



Normalisation using MinMax Scaling

In [35]:

```
## Feature Scaling
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Feature selection using SelectKBest

In [36]:

```
from sklearn.feature_selection import SelectKBest, f_regression

fs = SelectKBest(k=3)
X_train_scaled = fs.fit_transform(X_train_scaled, y_train)
X_test_scaled = fs.transform(X_test_scaled)
```

Model Selection

In [37]:
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import HuberRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
```

Creating the model for each regression technique

In [38]:
```
lr = LinearRegression().fit(X_train_scaled, y_train)
y_lr = lr.predict(X_test_scaled)
```

In [39]:
```
knn = KNeighborsRegressor(n_neighbors=3).fit(X_train_scaled, y_train)
y_knn = knn.predict(X_test_scaled)
```

In [40]:
```
dt = DecisionTreeRegressor().fit(X_train_scaled, y_train)
y_dt = dt.predict(X_test_scaled)
```

In [41]:
```
br = BayesianRidge().fit(X_train_scaled,y_train)
y_br = br.predict(X_test_scaled)
```

In [42]:
```
en = ElasticNet().fit(X_train_scaled,y_train)
y_en = en.predict(X_test_scaled)
```

In [43]:
```
gb = GradientBoostingRegressor().fit(X_train_scaled,y_train)
y_gb = gb.predict(X_test_scaled)
```

In [44]:
```
hr = HuberRegressor().fit(X_train_scaled,y_train)
y_hr = hr.predict(X_test_scaled)
```

In [45]:
```
svr = SVR().fit(X_train_scaled,y_train)
y_svr = svr.predict(X_test_scaled)
```

In [46]:
```
rf = RandomForestRegressor().fit(X_train_scaled,y_train)
y_rf = rf.predict(X_test_scaled)
```

In [47]:

```
et = ExtraTreesRegressor().fit(X_train_scaled,y_train)
y_et = et.predict(X_test_scaled)
```

Model Evaluation

R2 Score

Higher R2 Score is better

In [48]:

```
lr_score = metrics.r2_score(y_test, y_lr)
knn_score = metrics.r2_score(y_test, y_knn)
dt_score = metrics.r2_score(y_test, y_dt)
br_score = metrics.r2_score(y_test, y_br)
en_score = metrics.r2_score(y_test, y_en)
gb_score = metrics.r2_score(y_test, y_gb)
hr_score = metrics.r2_score(y_test, y_hr)
svr_score = metrics.r2_score(y_test, y_svr)
rf_score = metrics.r2_score(y_test, y_rf)
et_score = metrics.r2_score(y_test, y_et)
```

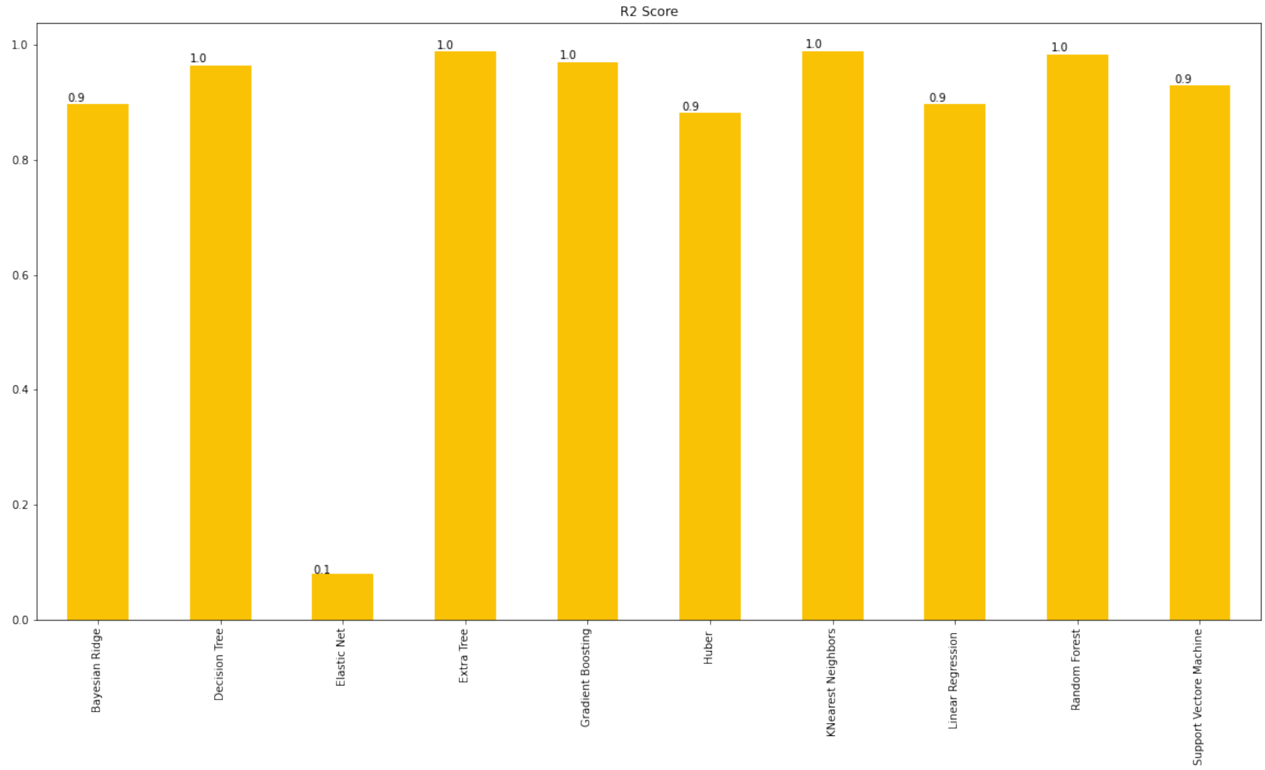
In [49]:

```
from sklearn.metrics import accuracy_score, classification_report, confusio

print("*"*20, "R2 Score", "*"*20)

print("-"*50)
print("| Linear Regression: ", lr_score)
print("-"*50)

print("-"*50)
print("| KNearest Neighbors: ", knn_score)
print("-"*50)

print("-"*50)
print("| Decision Tree: ", dt_score)
print("-"*50)

print("-"*50)
print("| Bayesian Ridge: ", br_score)
print("-"*50)

print("-"*50)
print("| Elastic Net: ", en_score)
print("-"*50)

print("-"*50)
print("| Gradient Boosting: ", gb_score)
print("-"*50)

print("-"*50)
print("| Huber: ", hr_score)
print("-"*50)

print("-"*50)
print("| Support Vector Machine: ", svr_score)
print("-"*50)

print("-"*50)
print("| Random Forest: ", rf_score)
print("-"*50)

print("-"*50)
print("| Extra Tree: ", et_score)
print("-"*50)
```

```
******************** R2 Score ********************
--------------------------------------------------
| Linear Regression:  0.8969155673669311
--------------------------------------------------
--------------------------------------------------
| KNearest Neighbors:  0.9886213810048285
--------------------------------------------------
--------------------------------------------------
| Decision Tree:  0.9640886277050201
--------------------------------------------------
--------------------------------------------------
| Bayesian Ridge:  0.8968993982916906
--------------------------------------------------
--------------------------------------------------
| Elastic Net:  0.080366806910785 79
--------------------------------------------------
--------------------------------------------------
| Gradient Boosting:  0.9703628591226864
--------------------------------------------------
--------------------------------------------------
| Huber:  0.8812285152187818
--------------------------------------------------
--------------------------------------------------
| Support Vector Machine:  0.9289393066490041
--------------------------------------------------
--------------------------------------------------
| Random Forest:  0.9833733204876863
--------------------------------------------------
--------------------------------------------------
| Extra Tree:  0.9885125775577236
--------------------------------------------------
```

In [57]:

```
metric_val = {
    "R2 score": {
    "Linear Regression ": lr_score,
    "KNearest Neighbors": knn_score,
    "Decision Tree": dt_score,
    "Bayesian Ridge": br_score,
    "Elastic Net": en_score,
    "Gradient Boosting": gb_score,
    "Huber ": hr_score,
    "Support Vectore Machine": svr_score,
    "Random Forest": rf_score,
    "Extra Tree": et_score
    }
}

ax = pd.DataFrame(metric_val).plot(kind="bar",
                             figsize = (20,10),
                             legend =False,
                             title = "R2 Score",
                             color = '#FAC205');

for p in ax.patches:
    ax.annotate(str(round(p.get_height(), 1)), (p.get_x() * 1.005, p.get_he
```



Mean Square Error

Lower MSE is better

In [51]:

```
lr_score_MSE = metrics.mean_squared_error(y_test, y_lr)
knn_score_MSE = metrics.mean_squared_error(y_test, y_knn)
dt_score_MSE = metrics.mean_squared_error(y_test, y_dt)
br_score_MSE = metrics.mean_squared_error(y_test, y_br)
en_score_MSE = metrics.mean_squared_error(y_test, y_en)
gb_score_MSE = metrics.mean_squared_error(y_test, y_gb)
hr_score_MSE = metrics.mean_squared_error(y_test, y_hr)
svr_score_MSE = metrics.mean_squared_error(y_test, y_svr)
rf_score_MSE = metrics.mean_squared_error(y_test, y_rf)
et_score_MSE = metrics.mean_squared_error(y_test, y_et)
```

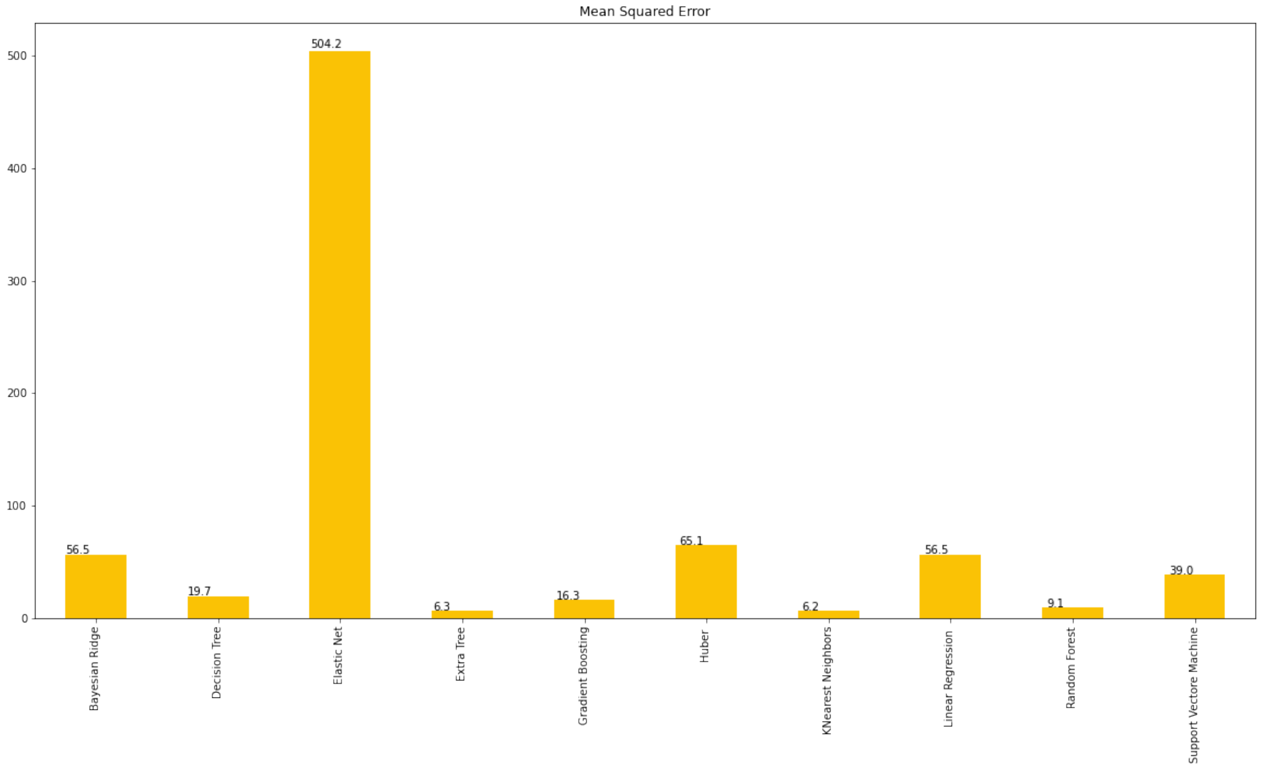
In [52]:

```
from sklearn.metrics import accuracy_score, classification_report, confusi

print("*"*20, "Mean Squared Error", "*"*20)

print("-"*50)
print("| Linear Regression: ", lr_score_MSE)
print("-"*50)

print("-"*50)
print("| KNearest Neighbors: ", knn_score_MSE)
print("-"*50)

print("-"*50)
print("| Decision Tree: ", dt_score_MSE)
print("-"*50)

print("-"*50)
print("| Bayesian Ridge: ", br_score_MSE)
print("-"*50)

print("-"*50)
print("| Elastic Net: ", en_score_MSE)
print("-"*50)

print("-"*50)
print("| Gradient Boosting: ", gb_score_MSE)
print("-"*50)

print("-"*50)
print("| Huber: ", hr_score_MSE)
print("-"*50)

print("-"*50)
print("| Support Vector Machine: ", svr_score_MSE)
print("-"*50)

print("-"*50)
print("| Random Forest: ", rf_score_MSE)
print("-"*50)

print("-"*50)
print("| Extra Tree: ", et_score_MSE)
print("-"*50)
```

```
******************* Mean Squared Error *******************
--------------------------------------------------
| Linear Regression:  56.52118365747226
--------------------------------------------------
--------------------------------------------------
| KNearest Neighbors:  6.238895607872545
--------------------------------------------------
--------------------------------------------------
| Decision Tree:  19.690201682550512
--------------------------------------------------
--------------------------------------------------
| Bayesian Ridge:  56.530049159739626
--------------------------------------------------
--------------------------------------------------
| Elastic Net:  504.23478382155474
--------------------------------------------------
--------------------------------------------------
| Gradient Boosting:  16.250041250861496
--------------------------------------------------
--------------------------------------------------
| Huber:  65.12239271360531
--------------------------------------------------
--------------------------------------------------
| Support Vector Machine:  38.962570750285195
--------------------------------------------------
--------------------------------------------------
| Random Forest:  9.116406641869066
--------------------------------------------------
--------------------------------------------------
| Extra Tree:  6.2985525265682805
--------------------------------------------------
```

In [53]:

```
metric_val = {
    "Mean Squared Error": {
    "Linear Regression ": lr_score_MSE,
    "KNearest Neighbors": knn_score_MSE,
    "Decision Tree": dt_score_MSE,
    "Bayesian Ridge": br_score_MSE,
    "Elastic Net": en_score_MSE,
    "Gradient Boosting": gb_score_MSE,
    "Huber ": hr_score_MSE,
    "Support Vectore Machine": svr_score_MSE,
    "Random Forest": rf_score_MSE,
    "Extra Tree": et_score_MSE
    }
}

ax = pd.DataFrame(metric_val).plot(kind="bar",
                             figsize = (20,10),
                             legend =False,
                             title = "Mean Squared Error",
                             color = '#FAC205');

for p in ax.patches:
    ax.annotate(str(round(p.get_height(), 1)), (p.get_x() * 1.005, p.get_he
```



Mean Absolute Error

Lower MAE is better

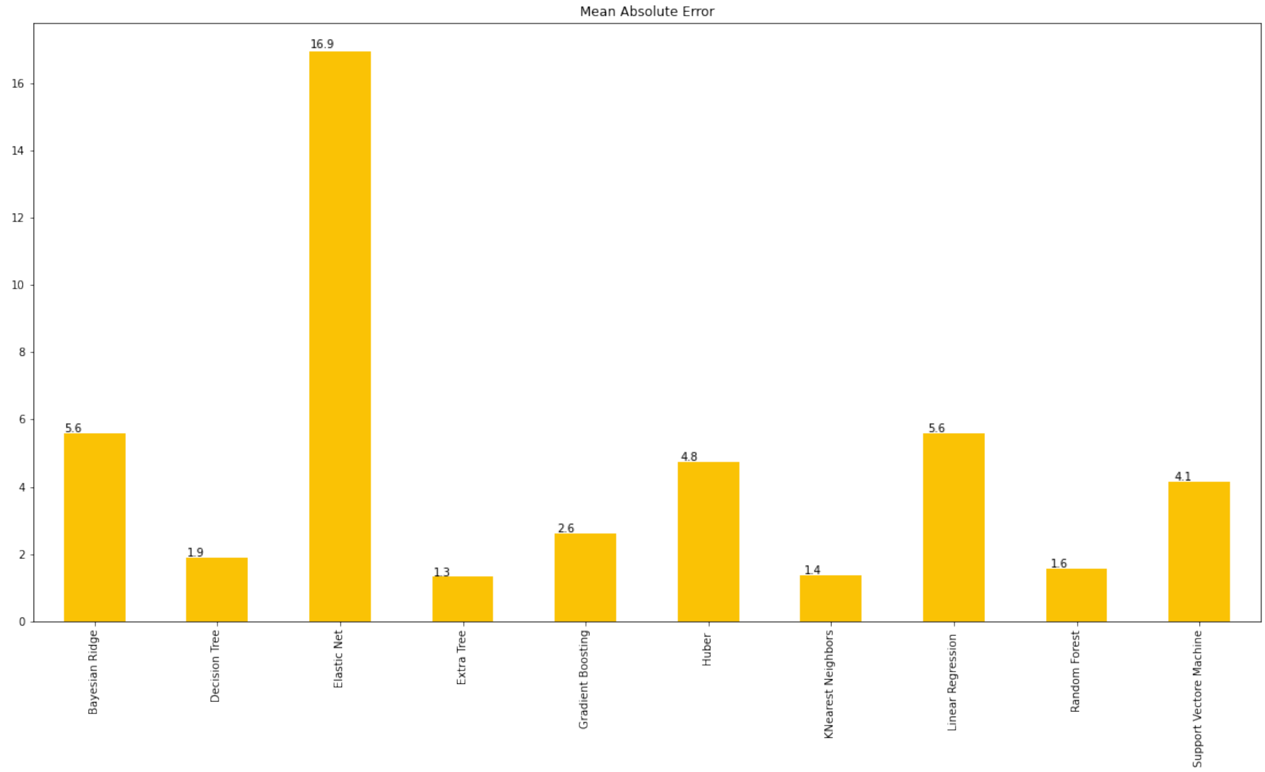
In [54]:
```
lr_score_MAE = metrics.mean_absolute_error(y_test, y_lr)
knn_score_MAE = metrics.mean_absolute_error(y_test, y_knn)
dt_score_MAE = metrics.mean_absolute_error(y_test, y_dt)
br_score_MAE = metrics.mean_absolute_error(y_test, y_br)
en_score_MAE = metrics.mean_absolute_error(y_test, y_en)
gb_score_MAE = metrics.mean_absolute_error(y_test, y_gb)
hr_score_MAE = metrics.mean_absolute_error(y_test, y_hr)
svr_score_MAE = metrics.mean_absolute_error(y_test, y_svr)
rf_score_MAE = metrics.mean_absolute_error(y_test, y_rf)
et_score_MAE = metrics.mean_absolute_error(y_test, y_et)
```

In [55]:
```
from sklearn.metrics import accuracy_score, classification_report, confusi

print("*"*20, "Mean Squared Error", "*"*20)

print("-"*50)
print("| Linear Regression: ", lr_score_MAE)
print("-"*50)

print("-"*50)
print("| KNearest Neighbors: ", knn_score_MAE)
print("-"*50)

print("-"*50)
print("| Decision Tree: ", dt_score_MAE)
print("-"*50)

print("-"*50)
print("| Bayesian Ridge: ", br_score_MAE)
print("-"*50)

print("-"*50)
print("| Elastic Net: ", en_score_MAE)
print("-"*50)

print("-"*50)
print("| Gradient Boosting: ", gb_score_MAE)
print("-"*50)

print("-"*50)
print("| Huber: ", hr_score_MSE)
print("-"*50)

print("-"*50)
print("| Support Vector Machine: ", svr_score_MAE)
print("-"*50)

print("-"*50)
print("| Random Forest: ", rf_score_MAE)
print("-"*50)

print("-"*50)
print("| Extra Tree: ", et_score_MAE)
print("-"*50)
```

```
******************* Mean Squared Error *******************
-------------------------------------------------
| Linear Regression:  5.599759825271187
-------------------------------------------------
-------------------------------------------------
| KNearest Neighbors:  1.3796944039301322
-------------------------------------------------
-------------------------------------------------
| Decision Tree:  1.8989516310043668
-------------------------------------------------
-------------------------------------------------
| Bayesian Ridge:  5.600753035966268
-------------------------------------------------
-------------------------------------------------
| Elastic Net:  16.940915874599266
-------------------------------------------------
-------------------------------------------------
| Gradient Boosting:  2.6188394414321783
-------------------------------------------------
-------------------------------------------------
| Huber:  65.12239271360531
-------------------------------------------------
-------------------------------------------------
| Support Vector Machine:  4.148974511947981
-------------------------------------------------
-------------------------------------------------
| Random Forest:  1.576058357860267
-------------------------------------------------
-------------------------------------------------
| Extra Tree:  1.3188307264628882
-------------------------------------------------
```

In [56]:

```
metric_val = {
    "Mean Absolute Error": {
    "Linear Regression ": lr_score_MAE,
    "KNearest Neighbors": knn_score_MAE,
    "Decision Tree": dt_score_MAE,
    "Bayesian Ridge": br_score_MAE,
    "Elastic Net": en_score_MAE,
    "Gradient Boosting": gb_score_MAE,
    "Huber ": hr_score_MAE,
    "Support Vectore Machine": svr_score_MAE,
    "Random Forest": rf_score_MAE,
    "Extra Tree": et_score_MAE
    }
}

ax = pd.DataFrame(metric_val).plot(kind="bar",
                             figsize = (20,10),
                             legend =False,
                             title = "Mean Absolute Error",
                             color = '#FAC205');

for p in ax.patches:
    ax.annotate(str(round(p.get_height(), 1)), (p.get_x() * 1.005, p.get_he
```



Selected Models

After comparing the above graphs we have decided to move ahead with the following models :

1. KNearest Neighbors
2. Random Forest
3. Extra Tree

KNN

In [73]:
```
y_knn
```

Out[73]:
```
array([122.32333367, 128.67666867, 127.94999933,  96.56      ,
       118.160001  , 115.45333333, 125.64333333, 117.443334  ,
       107.596667  ,  98.56999967,  97.100001  , 167.94999667,
       141.870005  , 117.993332  , 171.05000267,  85.373334  ,
       121.693334  , 108.16333   , 113.87333433, 131.01667267,
       125.22666667, 113.32999933, 115.78666667, 108.60666667,
       108.52000167, 126.459999  , 125.66666667, 114.81999967,
       113.323336  , 127.24999733, 148.58666967,  90.90666733,
       157.87666333, 115.46999867, 114.06      , 120.11333433,
       142.33000167, 161.25333633, 173.83333333, 152.71000133,
       117.33999867, 113.87333433, 121.549998  , 114.58666733,
       121.26666767, 107.94      ,  88.219999  , 115.45333333,
       128.67666867, 117.91333533,  99.383334  , 128.67666867,
       107.79333233, 160.31000267, 136.99333733, 116.74000033,
       143.28333533, 131.24000033,  94.866669  , 124.25666833,
       116.12666567,  87.613332  , 104.55666567, 113.44      ,
```

```
 84.08333333, 122.88333367, 116.28000133, 113.43666833,
168.84666933,  90.44      ,  79.81      , 160.12      ,
158.880005  , 108.72666633, 141.91666667, 109.55999733,
124.25666567, 128.11333733, 113.52333333, 120.373334  ,
138.11999533, 107.18      ,  94.03333567,  90.643331  ,
110.63      , 120.87333433, 109.32333367, 112.06      ,
169.87      , 162.599996  , 106.94666533, 123.67000067,
107.596667  , 116.933334  , 126.18333167, 107.896665  ,
153.580002  ,  84.83999867, 131.01667267, 113.323336  ,
159.74333167, 111.10333233, 111.11000067, 107.866666  ,
142.71333333,  89.40666733,  92.64333333, 175.17333467,
118.32666767, 118.68999967, 121.20333367, 172.04333467,
136.99333733, 119.33666733, 160.59000133, 118.77666467,
119.01666533, 110.11666633, 119.82999933, 122.43000033,
129.03999833, 114.69      ,  89.769999  , 116.06333433,
131.55999767, 114.196668  , 125.87666567,  89.83000167,
107.596667  , 116.890002  , 110.13666533, 166.87999967,
 81.06      , 123.33666467,  74.736669  , 110.95666533,
104.72333533, 123.846667  ,  77.49333467, 125.37      ,
120.010002  , 107.78333033,  90.42333233, 131.13999933,
142.44000267, 177.53333533, 125.676666  , 125.91333233,
122.37333433,  90.726667  , 149.77000433, 103.29999767,
117.246666  , 132.599996  , 135.95666533, 118.153333  ,
116.540001  , 102.58333067, 123.806666  ,  91.116666  ,
108.42333233, 116.62333433, 172.37      , 118.03666667,
116.076665  , 156.443334  , 110.373334  ,  87.903333  ,
116.55333467, 124.529999  , 120.933334  , 119.93333433,
 96.56      , 109.31999967, 115.59999833, 127.72000133,
156.52333567, 107.596667  , 123.649999  , 139.406667  ,
 91.25999967, 117.99333467, 128.453336  , 115.46999867,
109.32333367, 118.55666833, 126.83666733, 125.74666633,
149.96333333, 111.93000033,  94.12999967, 115.28000133,
126.63666567, 120.88000233, 121.660001  ,  93.32666767,
120.06666567,  90.726667  , 119.943334  , 124.64333333,
122.14666733, 133.81667033, 124.469999  , 115.20333333,
127.21666733, 112.386668  , 165.98000067, 123.356664  ,
120.13333667, 112.80999767, 120.16999833, 119.43333433,
105.886668  , 115.48000067, 125.87666567, 172.12333133,
 86.686666  , 133.06332933, 128.07999933,  72.290001  ,
119.23333233,  89.403333  , 144.893331  ,  92.64333333,
151.256668  , 102.266665  , 103.07333367, 102.62      ,
118.88999933, 164.95667   , 121.45666767, 135.32666533,
 97.06666567, 112.27999867, 132.59000133, 147.573334  ,
126.34000133, 103.29999767, 123.98333233, 160.75999967,
120.193334  , 126.93333433, 127.21666733, 115.866666  ,
156.976669  , 129.03999833, 115.04333233, 177.556666  ,
120.23666867, 120.846667  , 101.99000033, 161.27999867,
114.196668  , 118.56999967, 125.38666533, 116.92333467,
115.32333367,  91.73666667,  98.56999967, 131.726669  ,
118.75333667, 170.55999733, 109.79666633,  87.153333  ,
 91.646665  , 156.456665  , 148.813334  , 152.75333133,
 73.100001  , 120.95666767, 117.14333333, 159.313334  ,
135.95666533, 110.943334  , 110.40666733, 160.02666733,
125.        , 120.86000067, 118.75999967, 160.550003  ,
104.10666667,  89.39333367,  81.513331  ,  88.97333267,
115.32333367, 113.653333  , 119.22000133, 119.72666667,
 76.86666633,  89.62999967, 153.95333867, 119.55333433,
```

```
       132.46666967, 127.903333  , 114.30000067,  81.59000133,
       117.94666533,  88.55333467, 118.43333167, 163.24000033,
       121.84333533, 110.63      , 124.83999867, 113.290001  ,
       136.930003  ,  80.62333467, 162.983332  , 135.30666633,
       164.13666767, 128.07000233,  92.63666533, 109.79666633,
       115.04333233, 127.60000367, 119.84666933,  92.82      ,
       133.13999933, 162.813334  ,  71.853335  , 118.25      ,
       107.95333367, 114.639999  , 122.016668  , 111.469999  ,
       119.236669  , 118.753334  , 126.26666767, 123.67000067,
       109.99999733, 157.87666333, 166.813334  , 112.05666633,
       168.529999  , 110.89333367, 161.25333633, 126.50666567,
       167.873332  , 135.80333433, 110.12999733, 166.356664  ,
       115.26333367,  72.55666867, 113.25333133,  93.38666767,
        88.36000067, 104.113332  , 126.29666633, 123.356664  ,
       170.06666533, 120.91666667,  86.31999733, 130.88      ,
       121.846667  , 109.79666633, 170.443334  , 126.63999967,
       126.33666733, 113.25333133, 141.340001  , 125.723333  ,
       144.09666467, 123.18666567, 116.316668  , 122.95666733,
       167.29666633,  72.02666733, 162.98000067, 170.10000067,
       118.693334  , 104.47999833, 127.526665  , 152.71333333,
       172.12333133, 134.86333233, 127.69000233, 127.656667  ,
       158.04333   ,  89.39333367, 130.756668  , 108.56      ,
       166.87000533, 156.47000133, 169.72000133, 121.823334  ,
        89.846667  , 131.01667267,  98.233332  , 127.29333267,
       128.07999933, 110.13666533,  90.47333267, 153.82667033,
        96.100001  ,  87.613332  , 124.586665  ,  87.32333133,
        93.18666833, 113.25333133, 156.356669  , 146.656667  ,
       105.53333533, 167.40333033, 112.03333533, 128.51999933,
        90.82999933, 109.31666833,  78.173335  , 111.100001  ,
       165.649999  , 141.656667  , 150.700002  , 163.056666  ,
        91.09666433, 117.91333533,  92.87999967, 129.02666733,
       115.57000233, 116.92666867, 125.313334  , 120.08666467,
        97.06666567, 164.97666933, 150.45667   , 125.15999867,
       170.06666533,  84.06333433, 168.03666667, 128.67666867,
       119.56666833,  88.97333267, 119.77666467,  83.653333  ,
       118.88      , 109.17000067, 116.21333067, 148.45333367,
       132.95667   , 118.55666833, 118.52333033, 122.12000033,
       115.46666733, 118.56999967, 121.92333467, 144.23333233,
       165.57000233, 171.05000267,  97.673332  , 159.30999767,
        93.18666833, 140.91999833, 121.31666833,  85.373334  ,
       105.18      , 123.25      , 168.529999  ,  92.63666533,
        94.866669  , 154.470001  ])
```

In [75]:

```
y_test
```

Out[75]:

```
[122.32,
 129.899994,
 126.980003,
 96.5,
 117.580002,
 115.0,
 125.440002,
 116.93,
 108.220001,
 98.830002,
 96.910004,
```

```
168.789993,
151.029999,
115.839996,
169.809998,
85.129997,
122.639999,
107.849998,
110.449997,
131.240005,
124.940002,
115.379997,
116.650002,
109.25,
110.529999,
125.720001,
123.709999,
114.949997,
114.290001,
126.860001,
146.869995,
89.440002,
167.270004,
115.050003,
117.110001,
120.620003,
141.630005,
160.649994,
173.309998,
152.300003,
117.459999,
113.150002,
121.309998,
113.470001,
121.349998,
107.519997,
88.839996,
114.440002,
130.559998,
117.739998,
99.669998,
129.770004,
106.169998,
161.070007,
127.93,
115.940002,
143.470001,
130.110001,
95.730003,
124.360001,
116.620003,
85.599998,
104.099998,
112.610001,
86.519997,
122.400002,
116.470001,
112.660004,
```

```
166.399994,
91.989998,
80.809998,
160.559998,
157.639999,
103.419998,
135.020004,
110.400002,
124.43,
127.739998,
112.440002,
119.910004,
137.809998,
105.720001,
93.559998,
92.730003,
111.510002,
119.580002,
109.980003,
111.970001,
166.380005,
161.539993,
106.480003,
124.779999,
106.129997,
115.849998,
125.720001,
107.75,
162.009995,
78.389999,
129.520004,
115.639999,
145.729996,
109.790001,
112.290001,
107.339996,
136.179993,
86.449997,
92.059998,
178.539993,
118.93,
120.559998,
121.559998,
172.410004,
130.100006,
119.330002,
159.619995,
118.989998,
118.32,
110.239998,
119.190002,
122.07,
129.649994,
114.099998,
88.139999,
114.769997,
131.660004,
```

```
114.470001,
124.150002,
89.540001,
106.220001,
117.220001,
108.599998,
165.880005,
80.389999,
122.230003,
72.510002,
109.139999,
104.040001,
122.879997,
76.949997,
128.119995,
121.269997,
110.949997,
92.629997,
129.869995,
135.320007,
172.360001,
128.470001,
127.480003,
124.910004,
92.559998,
151.589996,
103.559998,
117.919998,
133.110001,
135.520004,
119.650002,
117.300003,
102.550003,
124.400002,
90.040001,
108.279999,
118.470001,
161.490005,
116.25,
116.120003,
157.460007,
111.43,
87.379997,
116.980003,
124.690002,
121.199997,
120.730003,
99.169998,
107.949997,
114.43,
126.449997,
156.479996,
107.839996,
124.230003,
139.410004,
91.110001,
117.870003,
```

```
127.400002,
114.870003,
109.43,
119.169998,
127.269997,
124.279999,
148.589996,
111.860001,
94.599998,
113.639999,
124.82,
120.870003,
122.669998,
93.129997,
119.940002,
93.269997,
121.489998,
123.919998,
122.129997,
132.130005,
123.760002,
115.059998,
127.970001,
112.220001,
164.860001,
122.900002,
120.099998,
112.75,
119.550003,
122.290001,
107.040001,
116.550003,
125.779999,
172.100006,
85.199997,
134.179993,
127.059998,
73.580002,
118.970001,
89.220001,
162.070007,
92.660004,
160.5,
102.279999,
102.040001,
103.110001,
118.080002,
166.070007,
120.360001,
136.5,
97.010002,
114.309998,
132.490005,
146.5,
126.300003,
101.760002,
124.720001,
```

```
157.210007,
122.379997,
127.279999,
128.070007,
116.330002,
157.320007,
130.270004,
113.970001,
179.100006,
119.699997,
121.050003,
102.940002,
160.520004,
115.050003,
118.519997,
126.089996,
116.5,
115.470001,
91.150002,
98.360001,
131.699997,
118.940002,
172.070007,
107.470001,
85.809998,
92.389999,
156.460007,
136.240005,
152.990005,
71.099998,
121.68,
118.190002,
159.050003,
135.229996,
111.419998,
111.830002,
159.429993,
124.529999,
120.839996,
114.82,
161.220001,
104.400002,
89.18,
78.860001,
89.849998,
115.68,
114.32,
120.760002,
119.610001,
76.199997,
88.25,
153.029999,
119.779999,
132.009995,
128.460007,
113.75,
80.93,
```

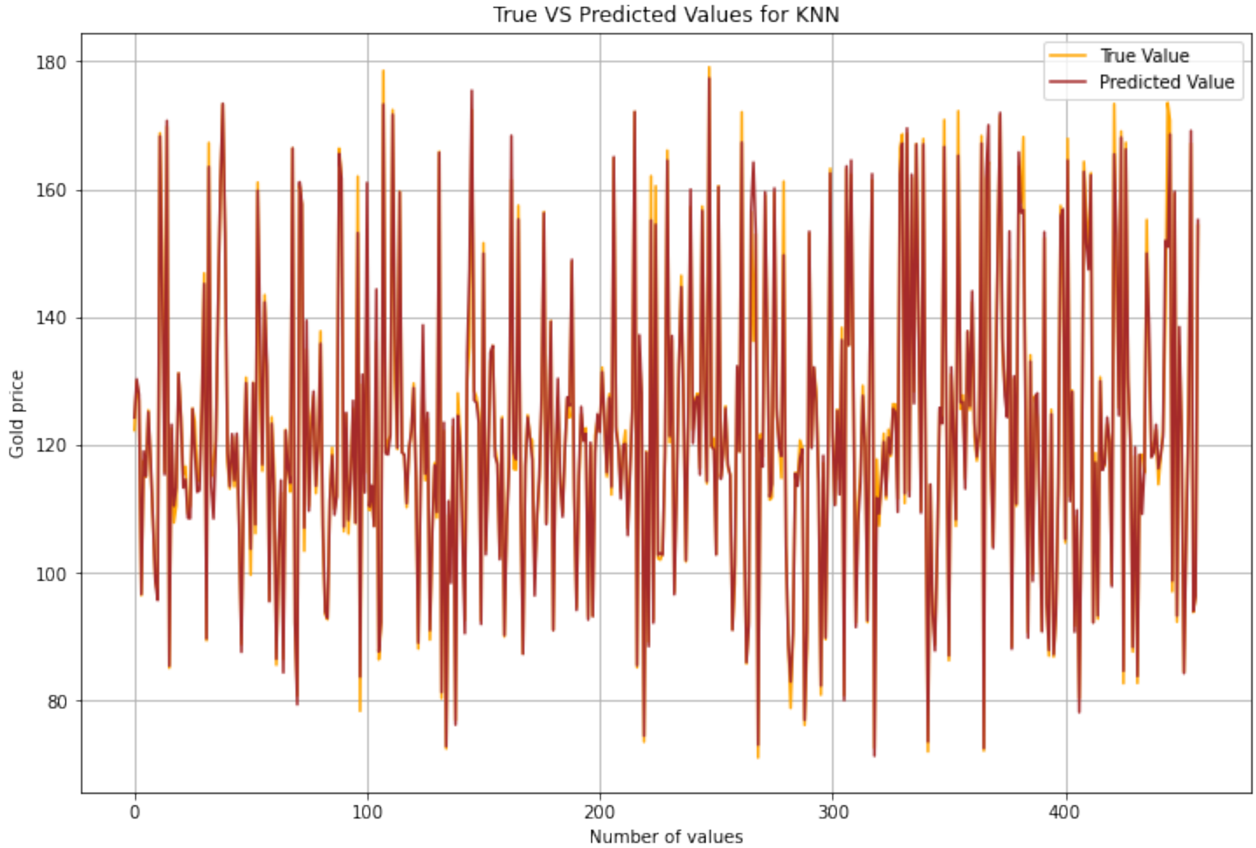
```
117.209999,
89.589996,
117.959999,
163.229996,
122.489998,
110.760002,
125.580002,
114.629997,
138.369995,
80.870003,
163.210007,
135.589996,
162.300003,
127.580002,
92.25,
111.019997,
115.269997,
129.289993,
119.699997,
92.269997,
133.690002,
161.419998,
72.650002,
117.769997,
107.370003,
115.110001,
121.800003,
111.540001,
122.419998,
118.230003,
126.419998,
126.339996,
111.589996,
166.419998,
168.610001,
110.889999,
168.710007,
112.849998,
161.839996,
128.380005,
167.119995,
137.660004,
109.800003,
167.869995,
115.889999,
72.050003,
113.779999,
95.120003,
88.800003,
104.389999,
125.32,
124.32,
170.850006,
120.860001,
86.309998,
131.029999,
122.699997,
```

```
107.309998,
172.229996,
125.620003,
127.779999,
114.720001,
137.779999,
125.459999,
142.380005,
121.290001,
117.519997,
121.650002,
168.350006,
72.18,
161.589996,
164.289993,
117.540001,
104.209999,
130.289993,
151.330002,
171.470001,
134.699997,
128.830002,
127.860001,
148.970001,
88.470001,
130.369995,
110.470001,
163.5,
157.929993,
168.179993,
122.709999,
90.709999,
134.070007,
99.82,
127.699997,
124.269997,
108.470001,
90.959999,
152.380005,
95.449997,
87.019997,
125.540001,
86.889999,
93.190002,
113.260002,
157.429993,
145.649994,
104.720001,
167.919998,
111.980003,
128.559998,
91.730003,
108.860001,
80.379997,
110.809998,
164.309998,
156.5,
```

```
149.740005,
162.559998,
92.290001,
118.769997,
92.790001,
130.619995,
117.339996,
116.790001,
123.32,
120.650002,
98.339996,
173.360001,
145.369995,
125.809998,
169.059998,
82.709999,
167.289993,
129.339996,
119.82,
87.68,
118.300003,
82.75,
118.459999,
118.489998,
115.620003,
155.229996,
133.139999,
118.150002,
119.800003,
122.419998,
113.830002,
118.18,
122.209999,
139.110001,
173.529999,
170.770004,
97.080002,
159.309998,
92.339996,
135.419998,
121.940002,
84.480003,
107.110001,
126.68,
167.179993,
93.849998,
96.230003,
154.339996]
```

In [91]:

```
# plot prediction VS original data
y_test = list(y_test)
plt.figure(figsize=(12, 8))
plt.plot(y_test, color = 'orange', label = 'True Value')
plt.plot(y_rf, color = 'brown', label = 'Predicted Value')
plt.legend()
plt.xlabel('Values')
plt.ylabel('Gold price')
plt.title('True VS Predicted Values for KNN')

plt.xlabel('Number of values')


plt.grid()
plt.show();
```



In [86]:

```
print ("The Accuracy of KNN model is :", knn_score*100,"%")
```

```
The Accuracy of KNN model is : 98.86213810048285 %
```

Random Forest

In [64]:

```
y_rf
```

Out[64]:

```
array([124.29410083, 130.30330324, 127.91790006,  96.66299689,
       119.03330046, 115.10349946, 125.18890079, 117.69589948,
       108.03130108,  98.7460997 ,  95.73319956, 168.29839733,
```

```
       144.03840143, 115.35750105, 170.71490177,  85.45700048,
       123.21429877, 110.4729972 , 113.28770144, 131.20550392,
       124.36769898, 113.33180082, 114.59440186, 108.57819928,
       108.52010215, 125.6684    , 119.00779983, 112.64349918,
       112.91890119, 125.83629878, 145.21440001,  89.71799978,
       163.51369961, 113.61709921, 108.48580067, 119.94210062,
       142.38180281, 161.13280009, 173.39749958, 153.1241007 ,
       118.90890117, 113.52420053, 121.73069838, 114.48449989,
       121.79250043, 107.61719988,  87.62519883, 114.92239929,
       129.70670202, 117.09950109, 103.74030004, 129.68860204,
       107.55689822, 159.78180317, 137.88500139, 116.94519966,
       142.34710195, 132.40280078,  95.51980137, 123.33730064,
       113.59179923,  86.54610149, 104.36639936, 114.47990012,
        84.4041997 , 122.27000093, 116.31429885, 114.03110205,
       166.43020209,  92.18920004,  79.41230046, 161.12510035,
       157.6066035 , 107.04950028, 139.48230215, 109.69049798,
       121.97050057, 128.39270135, 113.60069981, 120.19680139,
       135.85929757, 107.37250061,  93.90090127,  92.93029843,
       111.27940037, 118.38840007, 109.08770013, 112.05189947,
       165.57239785, 161.71329832, 107.20339872, 125.07110063,
       108.24019945, 114.368202  , 126.94689676, 107.83589898,
       153.16140035,  83.75029867, 131.03220403, 112.56030118,
       161.01719874, 110.40589835, 113.67939995, 107.26680031,
       144.35310053,  87.6818995 ,  92.35239923, 173.28390107,
       118.68280155, 118.49320011, 121.51130012, 171.67739916,
       137.0430013 , 119.5732994 , 159.56260231, 118.81489768,
       118.49699939, 110.87159924, 119.3837995 , 122.23539961,
       128.94829831, 115.0753999 ,  89.07169954, 114.88490099,
       138.75759792, 115.48630129, 125.05420017,  91.02730077,
       106.56720162, 116.80950164, 109.49679971, 165.72860196,
        81.32760019, 123.51689826,  72.8770008 , 111.23899921,
        98.4056012 , 124.07149989,  76.24589984, 124.61759938,
       119.86220081, 105.70549997,  90.55009924, 134.26440063,
       147.5454997 , 175.44459913, 126.9922997 , 126.62589881,
       123.43340074,  91.97629842, 149.96260044, 102.86389895,
       116.0624996 , 134.46879763, 135.51580035, 118.24770114,
       116.98850194, 102.0966987 , 124.05269854,  90.24549897,
       107.7600989 , 116.51720055, 168.38230033, 118.88339892,
       117.76459952, 155.29030238, 111.69310059,  87.31779885,
       116.62510117, 124.36169866, 120.95420228, 117.6358001 ,
        96.44919782, 108.89720021, 115.13359906, 127.93320145,
       156.28020113, 107.58499971, 124.18879929, 139.32440183,
        91.01820057, 118.74950077, 130.38870125, 114.01489877,
       108.76340001, 119.25850027, 127.49250076, 126.10759952,
       148.98690166, 112.64120122,  94.16439988, 115.24850033,
       125.99109986, 120.61580213, 121.76170043,  92.67710055,
       120.38669878,  93.19840019, 118.88760069, 124.8407001 ,
       121.99509934, 131.42170005, 124.14189874, 115.69570171,
       127.39060039, 113.44590045, 165.0508994 , 122.23469807,
       119.32930227, 111.58429937, 119.97959995, 120.21959963,
       105.87150063, 116.11640144, 125.75119827, 172.1120975 ,
        85.5512    , 137.20289767, 128.05020045,  74.51630071,
       118.87110061,  88.51529991, 155.14609997,  92.1738993 ,
       154.4816003 , 102.81299743, 103.08169959, 102.81679922,
       118.95449936, 164.50280234, 121.35880159, 137.0769982 ,
        96.61969786, 112.52849992, 132.65810049, 144.71539839,
       125.64490023, 101.95239892, 125.68690048, 159.97750016,
```

```
       120.29440222, 126.76940073, 127.66740255, 115.35559932,
       156.64930309, 128.50920084, 114.30929882, 177.35689923,
       119.91900195, 119.2160018 , 102.84299832, 160.38409988,
       114.67260088, 118.66229918, 125.76859994, 116.96930083,
       115.2138998 ,  91.05599951, 101.25149999, 132.34780125,
       118.97090202, 167.32329869, 108.80670128,  86.01550002,
        91.68489939, 156.6402983 , 164.20709976, 153.35050022,
        73.10539866, 120.80709979, 116.59940082, 159.56459892,
       135.44099995, 111.92519917, 113.97009939, 160.15740111,
       125.76730006, 119.68170139, 118.27629984, 149.69940316,
       104.0566995 ,  88.76700004,  82.9853994 ,  90.61089909,
       115.54600042, 113.58379943, 117.94650203, 119.32040052,
        76.99110034,  90.87820057, 153.39320242, 119.51840128,
       132.18320085, 126.59270169, 114.0584014 ,  82.34950065,
       118.3475987 ,  89.87640012, 117.29199966, 162.49550237,
       121.4995016 , 110.56450079, 125.36779914, 112.25259982,
       136.43219892,  80.07540064, 163.60609953, 135.59050116,
       164.52090143, 127.30800059,  91.50669886, 108.00099897,
       114.1153988 , 127.70330171, 119.74830091,  92.55459964,
       134.18669913, 162.38830055,  71.37910069, 111.76840009,
       109.28119937, 114.07779895, 120.52070249, 112.07919984,
       121.17149996, 118.61020198, 125.69930087, 125.13320045,
       109.51799987, 162.2770996 , 167.17189992, 112.40299976,
       169.53769841, 111.95440001, 162.30790225, 126.49459917,
       167.03489902, 137.7022992 , 109.37369842, 167.03199927,
       116.29840181,  73.59900065, 113.85180048,  93.6699001 ,
        87.82970085, 104.2477994 , 125.81490053, 123.34819807,
       166.62800027, 122.4530989 ,  87.06209862, 132.16779858,
       121.88810038, 108.32429956, 165.26869987, 126.62569823,
       126.76390025, 113.15180026, 137.82429963, 126.04989997,
       144.06500059, 123.37999951, 118.15120026, 122.11709984,
       167.17589923,  72.58430103, 162.69559856, 170.0361986 ,
       118.96470076, 103.87489819, 128.07149815, 150.61630055,
       171.95370191, 136.75669749, 127.74640324, 124.34300009,
       153.42089757,  88.10160033, 130.78730204, 110.7911009 ,
       165.78410046, 156.31629927, 156.75510019, 121.73840052,
        89.84929992, 133.07960124,  98.70799857, 127.37169879,
       128.10529889, 107.82919934,  90.81359857, 153.30980064,
        95.22529879,  87.90409908, 124.83379949,  87.25619785,
        93.68400082, 114.08570023, 155.78030291, 156.89449934,
       105.29290066, 164.52239864, 111.20240042, 128.30870168,
        90.75280007, 109.82129962,  78.19300034, 111.25179996,
       162.74509937, 151.61079988, 147.425801  , 162.20109875,
        92.20489822, 117.2881018 ,  93.30850104, 130.01270036,
       116.08170062, 117.18970059, 124.3155005 , 120.3608996 ,
        97.88199922, 165.49020145, 152.89840116, 124.6150988 ,
       168.10139733,  84.68300041, 166.25209924, 130.45860291,
       119.88260212,  88.3929001 , 119.667299  ,  83.85479942,
       118.50360068, 109.20020102, 116.44449856, 149.98649966,
       138.75990469, 118.08950087, 118.77229718, 123.18549876,
       116.30470089, 118.47720014, 121.35900113, 151.97030032,
       150.95999737, 168.58780151,  98.76099924, 159.63529816,
        93.29580065, 138.47599803, 121.39000089,  84.29669906,
       106.37649959, 124.0418991 , 169.15309788,  94.01159958,
        96.38750066, 155.17180164])
```

In [65]:

```
y_test
```

Out[65]:

```
[122.32,
 129.899994,
 126.980003,
 96.5,
 117.580002,
 115.0,
 125.440002,
 116.93,
 108.220001,
 98.830002,
 96.910004,
 168.789993,
 151.029999,
 115.839996,
 169.809998,
 85.129997,
 122.639999,
 107.849998,
 110.449997,
 131.240005,
 124.940002,
 115.379997,
 116.650002,
 109.25,
 110.529999,
 125.720001,
 123.709999,
 114.949997,
 114.290001,
 126.860001,
 146.869995,
 89.440002,
 167.270004,
 115.050003,
 117.110001,
 120.620003,
 141.630005,
 160.649994,
 173.309998,
 152.300003,
 117.459999,
 113.150002,
 121.309998,
 113.470001,
 121.349998,
 107.519997,
 88.839996,
 114.440002,
 130.559998,
 117.739998,
 99.669998,
 129.770004,
 106.169998,
```

```
161.070007,
127.93,
115.940002,
143.470001,
130.110001,
95.730003,
124.360001,
116.620003,
85.599998,
104.099998,
112.610001,
86.519997,
122.400002,
116.470001,
112.660004,
166.399994,
91.989998,
80.809998,
160.559998,
157.639999,
103.419998,
135.020004,
110.400002,
124.43,
127.739998,
112.440002,
119.910004,
137.809998,
105.720001,
93.559998,
92.730003,
111.510002,
119.580002,
109.980003,
111.970001,
166.380005,
161.539993,
106.480003,
124.779999,
106.129997,
115.849998,
125.720001,
107.75,
162.009995,
78.389999,
129.520004,
115.639999,
145.729996,
109.790001,
112.290001,
107.339996,
136.179993,
86.449997,
92.059998,
178.539993,
118.93,
120.559998,
```

```
121.559998,
172.410004,
130.100006,
119.330002,
159.619995,
118.989998,
118.32,
110.239998,
119.190002,
122.07,
129.649994,
114.099998,
88.139999,
114.769997,
131.660004,
114.470001,
124.150002,
89.540001,
106.220001,
117.220001,
108.599998,
165.880005,
80.389999,
122.230003,
72.510002,
109.139999,
104.040001,
122.879997,
76.949997,
128.119995,
121.269997,
110.949997,
92.629997,
129.869995,
135.320007,
172.360001,
128.470001,
127.480003,
124.910004,
92.559998,
151.589996,
103.559998,
117.919998,
133.110001,
135.520004,
119.650002,
117.300003,
102.550003,
124.400002,
90.040001,
108.279999,
118.470001,
161.490005,
116.25,
116.120003,
157.460007,
111.43,
```

```
87.379997,
116.980003,
124.690002,
121.199997,
120.730003,
99.169998,
107.949997,
114.43,
126.449997,
156.479996,
107.839996,
124.230003,
139.410004,
91.110001,
117.870003,
127.400002,
114.870003,
109.43,
119.169998,
127.269997,
124.279999,
148.589996,
111.860001,
94.599998,
113.639999,
124.82,
120.870003,
122.669998,
93.129997,
119.940002,
93.269997,
121.489998,
123.919998,
122.129997,
132.130005,
123.760002,
115.059998,
127.970001,
112.220001,
164.860001,
122.900002,
120.099998,
112.75,
119.550003,
122.290001,
107.040001,
116.550003,
125.779999,
172.100006,
85.199997,
134.179993,
127.059998,
73.580002,
118.970001,
89.220001,
162.070007,
92.660004,
```

```
160.5,
102.279999,
102.040001,
103.110001,
118.080002,
166.070007,
120.360001,
136.5,
97.010002,
114.309998,
132.490005,
146.5,
126.300003,
101.760002,
124.720001,
157.210007,
122.379997,
127.279999,
128.070007,
116.330002,
157.320007,
130.270004,
113.970001,
179.100006,
119.699997,
121.050003,
102.940002,
160.520004,
115.050003,
118.519997,
126.089996,
116.5,
115.470001,
91.150002,
98.360001,
131.699997,
118.940002,
172.070007,
107.470001,
85.809998,
92.389999,
156.460007,
136.240005,
152.990005,
71.099998,
121.68,
118.190002,
159.050003,
135.229996,
111.419998,
111.830002,
159.429993,
124.529999,
120.839996,
114.82,
161.220001,
104.400002,
```

```
89.18,
78.860001,
89.849998,
115.68,
114.32,
120.760002,
119.610001,
76.199997,
88.25,
153.029999,
119.779999,
132.009995,
128.460007,
113.75,
80.93,
117.209999,
89.589996,
117.959999,
163.229996,
122.489998,
110.760002,
125.580002,
114.629997,
138.369995,
80.870003,
163.210007,
135.589996,
162.300003,
127.580002,
92.25,
111.019997,
115.269997,
129.289993,
119.699997,
92.269997,
133.690002,
161.419998,
72.650002,
117.769997,
107.370003,
115.110001,
121.800003,
111.540001,
122.419998,
118.230003,
126.419998,
126.339996,
111.589996,
166.419998,
168.610001,
110.889999,
168.710007,
112.849998,
161.839996,
128.380005,
167.119995,
137.660004,
```

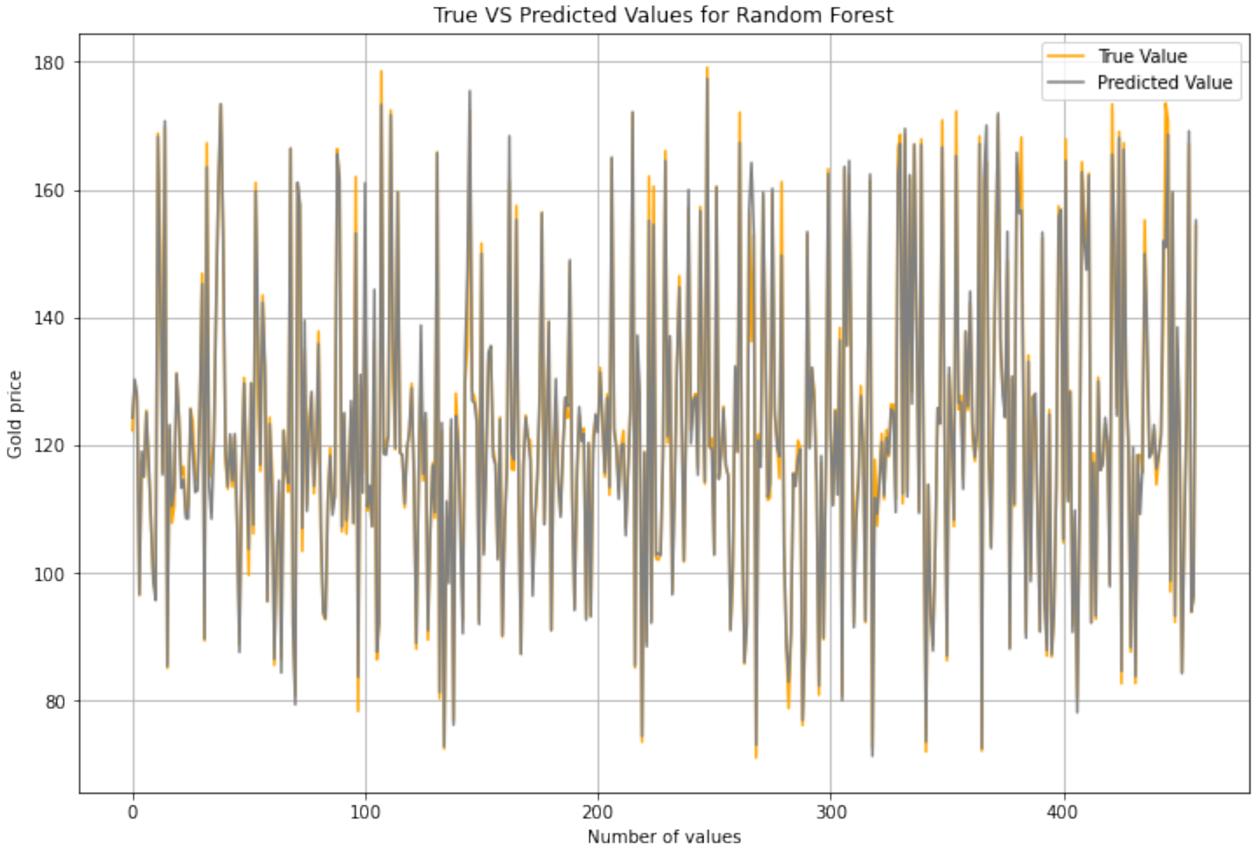
```
109.800003,
167.869995,
115.889999,
72.050003,
113.779999,
95.120003,
88.800003,
104.389999,
125.32,
124.32,
170.850006,
120.860001,
86.309998,
131.029999,
122.699997,
107.309998,
172.229996,
125.620003,
127.779999,
114.720001,
137.779999,
125.459999,
142.380005,
121.290001,
117.519997,
121.650002,
168.350006,
72.18,
161.589996,
164.289993,
117.540001,
104.209999,
130.289993,
151.330002,
171.470001,
134.699997,
128.830002,
127.860001,
148.970001,
88.470001,
130.369995,
110.470001,
163.5,
157.929993,
168.179993,
122.709999,
90.709999,
134.070007,
99.82,
127.699997,
124.269997,
108.470001,
90.959999,
152.380005,
95.449997,
87.019997,
125.540001,
```

```
86.889999,
93.190002,
113.260002,
157.429993,
145.649994,
104.720001,
167.919998,
111.980003,
128.559998,
91.730003,
108.860001,
80.379997,
110.809998,
164.309998,
156.5,
149.740005,
162.559998,
92.290001,
118.769997,
92.790001,
130.619995,
117.339996,
116.790001,
123.32,
120.650002,
98.339996,
173.360001,
145.369995,
125.809998,
169.059998,
82.709999,
167.289993,
129.339996,
119.82,
87.68,
118.300003,
82.75,
118.459999,
118.489998,
115.620003,
155.229996,
133.139999,
118.150002,
119.800003,
122.419998,
113.830002,
118.18,
122.209999,
139.110001,
173.529999,
170.770004,
97.080002,
159.309998,
92.339996,
135.419998,
121.940002,
84.480003,
```

```
107.110001,
126.68,
167.179993,
93.849998,
96.230003,
154.339996]
```

In [94]:

```
# plot prediction VS original data
y_test = list(y_test)
plt.figure(figsize=(12, 8))
plt.plot(y_test, color = 'orange', label = 'True Value')
plt.plot(y_rf, color = 'grey', label = 'Predicted Value')
plt.legend()
plt.xlabel('Values')
plt.ylabel('Gold price')
plt.title('True VS Predicted Values for Random Forest')

plt.xlabel('Number of values')


plt.grid()
plt.show();
```



Model Accuracy

In [85]:

```
print ("The Accuracy of the Random Forest model is :",rf_score*100,"%")
```

```
The Accuracy of the Random Forest model is : 98.33733204876863 %
```

Extra Tree

In [78]:
```
y_et
```

Out[78]:
```
array([123.54379998, 129.15620396, 128.70160076,  96.3032964 ,
       117.8226008 , 115.01649894, 125.54100001, 117.65429874,
       108.09050056, 102.11179963,  96.59360074, 168.48839687,
       144.56380228, 116.28120065, 170.87710188,  85.80830183,
       123.11079887, 108.0955968 , 113.20110075, 131.23750414,
       124.73119912, 113.26770048, 114.2234005 , 108.48829895,
       109.50380135, 125.84699901, 124.28859986, 112.48099916,
       112.94090168, 125.35579865, 146.38420375,  89.79759981,
       159.01809968, 113.602399  , 111.40850038, 120.07990083,
       142.53850328, 161.00690057, 173.37489993, 153.02260063,
       118.09680056, 113.86000013, 121.84629812, 114.36419979,
       121.6118005 , 107.57579998,  88.01729868, 114.6693988 ,
       129.32060233, 117.53120113, 102.71740052, 129.25050251,
       107.2798984 , 159.84060296, 138.12810156, 117.58669981,
       143.37820165, 131.01460116,  95.63580129, 123.99180166,
       115.00229912,  86.48350108, 104.68929871, 113.41989977,
        84.67589979, 122.35690104, 116.43569924, 112.78490213,
       165.62310082,  92.13990053,  80.27880069, 160.51159985,
       158.97950341, 108.03569958, 135.47100228, 110.21719734,
       123.47580092, 128.30930288, 113.33569912, 120.12470137,
       136.23479669, 107.51330094,  93.80210088,  92.65419809,
       110.90390002, 118.30500118, 109.06279992, 111.95689944,
       165.75229787, 162.199397  , 107.01189911, 124.81000013,
       107.3681006 , 115.07980184, 126.04749805, 108.16359792,
       159.20210114,  82.31099927, 130.883105  , 112.92500141,
       158.0845988 , 110.25559804, 113.3946998 , 107.49589933,
       141.79840067,  87.57849875,  92.37239905, 175.48230082,
       118.79100166, 118.08810012, 121.46780041, 171.18989991,
       135.40820093, 119.39859971, 160.31590079, 119.02659879,
       118.69309883, 110.60429966, 119.57039961, 122.33149927,
       129.16269735, 115.11710028,  89.64489947, 115.05180152,
       136.16089787, 115.27940114, 123.78460043,  90.70490136,
       106.70100145, 116.50840187, 109.01439926, 165.28170064,
        80.91920073, 123.37309893,  74.02000095, 111.29630028,
       100.11380143, 123.87499989,  76.02590007, 124.19079861,
       119.82500062, 108.08219964,  90.91999904, 135.88650042,
       145.80879845, 176.61639887, 126.73799952, 126.8788985 ,
       123.6861    ,  91.95389927, 149.68720048, 103.38719942,
       115.80989938, 134.94609703, 135.77040029, 118.56670092,
       117.16730155, 102.30229902, 124.2124988 ,  90.08569931,
       108.4655988 , 116.33720084, 165.24570204, 118.30440042,
       117.89499956, 156.04060263, 111.67860059,  86.9668989 ,
       116.05100091, 124.32319893, 121.13710194, 118.26710055,
        96.15369702, 109.02680027, 114.98319914, 127.38390111,
       156.29570131, 108.13580107, 124.2359988 , 138.99080254,
        90.49369886, 118.49830102, 129.88510103, 114.35709812,
       108.60879988, 119.40980032, 127.34600181, 125.23810113,
       149.10850174, 112.71430015,  94.1108995 , 115.60990047,
       126.01099968, 121.30270227, 121.83430024,  92.68450083,
       120.03119944,  93.06680072, 119.41120057, 124.31530091,
       121.81129925, 130.82289987, 124.11639893, 115.29790108,
```

```
127.54130133, 112.81950051, 165.75319973, 122.57799747,
118.82810156, 112.75009928, 119.96339976, 120.06520009,
105.9940001 , 116.49250106, 125.71519806, 172.08119856,
 85.98520034, 138.42809756, 127.81010082,  74.5525007 ,
119.23390073,  87.83669977, 159.71179955,  92.20489924,
155.9634004 , 102.72589757, 103.37509971, 102.94489948,
118.32879882, 164.68770172, 121.9962015 , 135.62259706,
 96.64429755, 112.69940017, 133.0705004 , 145.33300022,
126.52740092, 102.04229877, 125.54300053, 159.92999993,
120.48440217, 127.02190118, 127.64250399, 115.3678989 ,
156.79950214, 128.38579986, 114.37599872, 177.45459867,
119.73500175, 120.39920093, 102.63839802, 160.35720014,
114.47610079, 118.31869898, 125.70869987, 116.94080101,
115.24119952,  91.1793994 , 101.45549906, 132.05149979,
119.05390214, 168.16159817, 109.24600127,  86.48980056,
 91.60199923, 157.05329834, 158.87419975, 153.18540009,
 72.34169789, 120.34400072, 117.5641002 , 159.34500002,
135.75929964, 112.36689952, 113.93029972, 159.89440002,
125.33940084, 120.29800122, 118.5831    , 154.92820337,
104.31659938,  89.48249954,  80.17780057,  90.10969903,
115.33770064, 113.73689918, 117.95360252, 119.1409004 ,
 76.91700084,  90.67720042, 153.65320321, 119.19820116,
131.94349903, 129.01070112, 114.20560147,  81.78610175,
117.64489764,  89.82859988, 117.07899838, 162.58680354,
122.2943014 , 110.29740076, 125.32599848, 112.65150019,
136.9589012 ,  80.91440071, 164.03059889, 135.88210059,
163.67120083, 127.64189977,  91.84859887, 108.17409929,
114.01509843, 128.18010267, 119.40410212,  92.51259881,
134.27379812, 162.19530041,  70.66320033, 114.92850006,
107.98940004, 113.97599817, 120.73110184, 112.08630043,
121.16249977, 118.48120231, 126.0485    , 125.49180151,
109.3634994 , 157.85630038, 167.45640006, 112.30679988,
169.24750045, 111.87140106, 161.78920335, 126.84169867,
165.41199841, 137.21690058, 109.8888986 , 166.6897994 ,
116.04630191,  73.25260074, 113.91850048,  93.65440003,
 87.94750172, 104.58649912, 126.43710118, 123.77869753,
167.04239876, 121.83670007,  87.0372987 , 131.54829796,
121.77389942, 108.1748996 , 166.87460146, 126.11019764,
126.99810113, 113.47280089, 135.68000031, 125.90920033,
142.91089837, 123.82489948, 119.05589996, 122.90879959,
167.33379836,  71.77560047, 163.52639857, 167.11559828,
118.01689967, 103.66379764, 128.22409873, 148.9063013 ,
171.82240169, 136.89019756, 127.32500359, 126.38250059,
152.71929697,  88.6806001 , 130.73070144, 110.10400045,
165.73320175, 156.83759984, 167.44040172, 121.8729    ,
 89.86160011, 132.92170252,  98.23359806, 127.41509989,
127.88589997, 108.50349945,  90.83449802, 153.14269999,
 95.62739851,  87.40209771, 124.96119931,  87.59269743,
 93.19680124, 113.86620034, 155.67170199, 151.29289862,
105.87440143, 165.60259506, 111.26110061, 127.82600238,
 90.81620015, 109.4633002 ,  79.0959002 , 111.55059966,
163.61079696, 150.58249998, 149.66480203, 162.48059665,
 92.13789867, 117.44220161,  92.89740145, 129.43700078,
115.89790079, 116.96400093, 123.44990096, 120.31999954,
 98.2434989 , 169.35140239, 149.28830288, 124.90119887,
167.05029637,  84.46899992, 166.96349684, 129.635904  ,
119.49300186,  88.82840022, 119.54419973,  84.23500045,
```

```
       118.22300248, 109.34960064, 116.18559821, 150.4602004 ,
       136.71980412, 118.32530094, 118.78849729, 122.83539891,
       115.69680111, 118.42500002, 121.33030125, 149.97789978,
       151.02499937, 169.77330008,  98.44779907, 159.44069761,
        93.08590046, 141.01889833, 121.69790108,  84.51229958,
       105.7828989 , 124.38439952, 169.34789453,  93.29729987,
        96.04970116, 154.86970185])
```

In [79]:

```
y_test
```

Out[79]:

```
[122.32,
 129.899994,
 126.980003,
 96.5,
 117.580002,
 115.0,
 125.440002,
 116.93,
 108.220001,
 98.830002,
 96.910004,
 168.789993,
 151.029999,
 115.839996,
 169.809998,
 85.129997,
 122.639999,
 107.849998,
 110.449997,
 131.240005,
 124.940002,
 115.379997,
 116.650002,
 109.25,
 110.529999,
 125.720001,
 123.709999,
 114.949997,
 114.290001,
 126.860001,
 146.869995,
 89.440002,
 167.270004,
 115.050003,
 117.110001,
 120.620003,
 141.630005,
 160.649994,
 173.309998,
 152.300003,
 117.459999,
 113.150002,
 121.309998,
 113.470001,
 121.349998,
 107.519997,
```

```
88.839996,
114.440002,
130.559998,
117.739998,
99.669998,
129.770004,
106.169998,
161.070007,
127.93,
115.940002,
143.470001,
130.110001,
95.730003,
124.360001,
116.620003,
85.599998,
104.099998,
112.610001,
86.519997,
122.400002,
116.470001,
112.660004,
166.399994,
91.989998,
80.809998,
160.559998,
157.639999,
103.419998,
135.020004,
110.400002,
124.43,
127.739998,
112.440002,
119.910004,
137.809998,
105.720001,
93.559998,
92.730003,
111.510002,
119.580002,
109.980003,
111.970001,
166.380005,
161.539993,
106.480003,
124.779999,
106.129997,
115.849998,
125.720001,
107.75,
162.009995,
78.389999,
129.520004,
115.639999,
145.729996,
109.790001,
112.290001,
```

```
107.339996,
136.179993,
86.449997,
92.059998,
178.539993,
118.93,
120.559998,
121.559998,
172.410004,
130.100006,
119.330002,
159.619995,
118.989998,
118.32,
110.239998,
119.190002,
122.07,
129.649994,
114.099998,
88.139999,
114.769997,
131.660004,
114.470001,
124.150002,
89.540001,
106.220001,
117.220001,
108.599998,
165.880005,
80.389999,
122.230003,
72.510002,
109.139999,
104.040001,
122.879997,
76.949997,
128.119995,
121.269997,
110.949997,
92.629997,
129.869995,
135.320007,
172.360001,
128.470001,
127.480003,
124.910004,
92.559998,
151.589996,
103.559998,
117.919998,
133.110001,
135.520004,
119.650002,
117.300003,
102.550003,
124.400002,
90.040001,
```

```
108.279999,
118.470001,
161.490005,
116.25,
116.120003,
157.460007,
111.43,
87.379997,
116.980003,
124.690002,
121.199997,
120.730003,
99.169998,
107.949997,
114.43,
126.449997,
156.479996,
107.839996,
124.230003,
139.410004,
91.110001,
117.870003,
127.400002,
114.870003,
109.43,
119.169998,
127.269997,
124.279999,
148.589996,
111.860001,
94.599998,
113.639999,
124.82,
120.870003,
122.669998,
93.129997,
119.940002,
93.269997,
121.489998,
123.919998,
122.129997,
132.130005,
123.760002,
115.059998,
127.970001,
112.220001,
164.860001,
122.900002,
120.099998,
112.75,
119.550003,
122.290001,
107.040001,
116.550003,
125.779999,
172.100006,
85.199997,
```

```
134.179993,
127.059998,
73.580002,
118.970001,
89.220001,
162.070007,
92.660004,
160.5,
102.279999,
102.040001,
103.110001,
118.080002,
166.070007,
120.360001,
136.5,
97.010002,
114.309998,
132.490005,
146.5,
126.300003,
101.760002,
124.720001,
157.210007,
122.379997,
127.279999,
128.070007,
116.330002,
157.320007,
130.270004,
113.970001,
179.100006,
119.699997,
121.050003,
102.940002,
160.520004,
115.050003,
118.519997,
126.089996,
116.5,
115.470001,
91.150002,
98.360001,
131.699997,
118.940002,
172.070007,
107.470001,
85.809998,
92.389999,
156.460007,
136.240005,
152.990005,
71.099998,
121.68,
118.190002,
159.050003,
135.229996,
111.419998,
```

```
              111.830002,
              159.429993,
              124.529999,
              120.839996,
              114.82,
              161.220001,
              104.400002,
              89.18,
              78.860001,
              89.849998,
              115.68,
              114.32,
              120.760002,
              119.610001,
              76.199997,
              88.25,
              153.029999,
              119.779999,
              132.009995,
              128.460007,
              113.75,
              80.93,
              117.209999,
              89.589996,
              117.959999,
              163.229996,
              122.489998,
              110.760002,
              125.580002,
              114.629997,
              138.369995,
              80.870003,
              163.210007,
              135.589996,
              162.300003,
              127.580002,
              92.25,
              111.019997,
              115.269997,
              129.289993,
              119.699997,
              92.269997,
              133.690002,
              161.419998,
              72.650002,
              117.769997,
              107.370003,
              115.110001,
              121.800003,
              111.540001,
              122.419998,
              118.230003,
              126.419998,
              126.339996,
              111.589996,
              166.419998,
              168.610001,
```

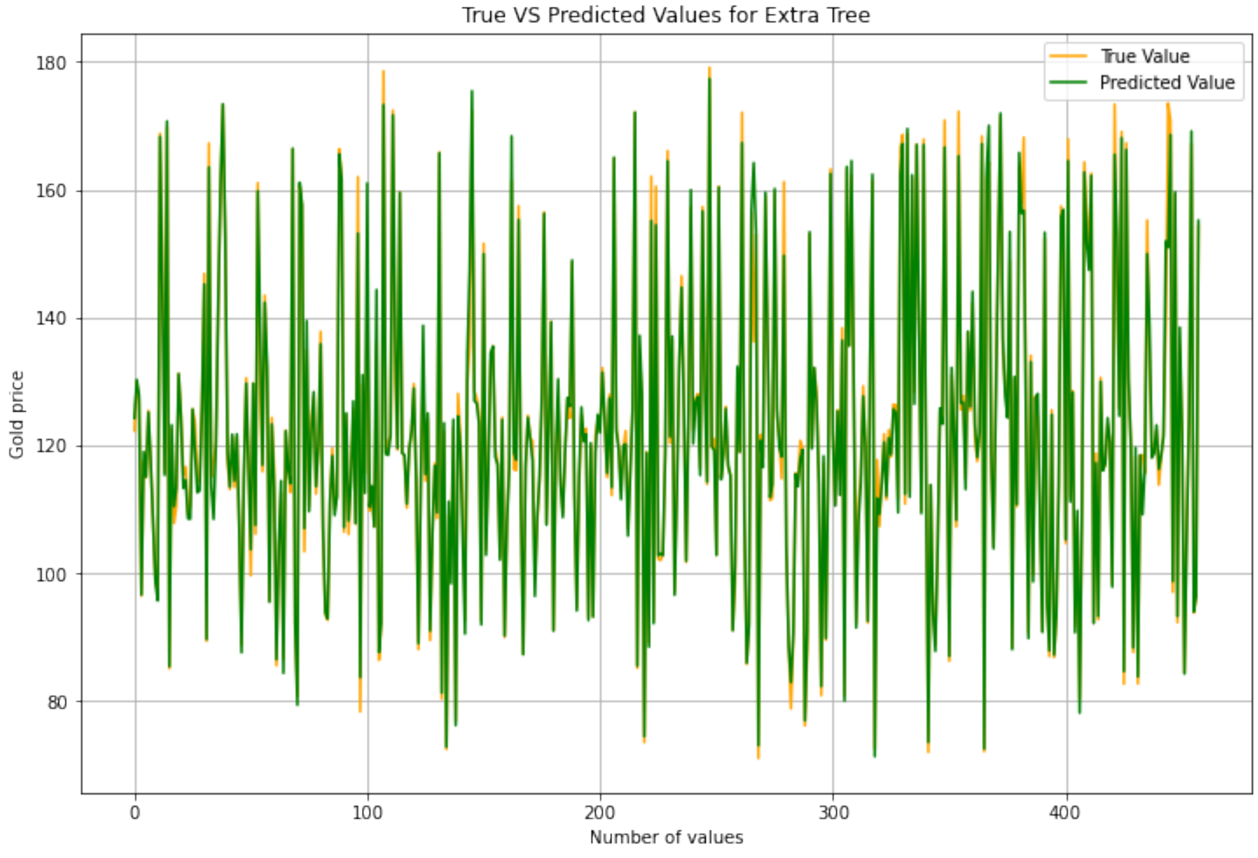
```
110.889999,
168.710007,
112.849998,
161.839996,
128.380005,
167.119995,
137.660004,
109.800003,
167.869995,
115.889999,
72.050003,
113.779999,
95.120003,
88.800003,
104.389999,
125.32,
124.32,
170.850006,
120.860001,
86.309998,
131.029999,
122.699997,
107.309998,
172.229996,
125.620003,
127.779999,
114.720001,
137.779999,
125.459999,
142.380005,
121.290001,
117.519997,
121.650002,
168.350006,
72.18,
161.589996,
164.289993,
117.540001,
104.209999,
130.289993,
151.330002,
171.470001,
134.699997,
128.830002,
127.860001,
148.970001,
88.470001,
130.369995,
110.470001,
163.5,
157.929993,
168.179993,
122.709999,
90.709999,
134.070007,
99.82,
127.699997,
```

```
124.269997,
108.470001,
90.959999,
152.380005,
95.449997,
87.019997,
125.540001,
86.889999,
93.190002,
113.260002,
157.429993,
145.649994,
104.720001,
167.919998,
111.980003,
128.559998,
91.730003,
108.860001,
80.379997,
110.809998,
164.309998,
156.5,
149.740005,
162.559998,
92.290001,
118.769997,
92.790001,
130.619995,
117.339996,
116.790001,
123.32,
120.650002,
98.339996,
173.360001,
145.369995,
125.809998,
169.059998,
82.709999,
167.289993,
129.339996,
119.82,
87.68,
118.300003,
82.75,
118.459999,
118.489998,
115.620003,
155.229996,
133.139999,
118.150002,
119.800003,
122.419998,
113.830002,
118.18,
122.209999,
139.110001,
173.529999,
```

```
170.770004,
97.080002,
159.309998,
92.339996,
135.419998,
121.940002,
84.480003,
107.110001,
126.68,
167.179993,
93.849998,
96.230003,
154.339996]
```

In [88]:

```python
# plot prediction VS original data
y_test = list(y_test)
plt.figure(figsize=(12, 8))
plt.plot(y_test, color = 'orange', label = 'True Value')
plt.plot(y_rf, color = 'green', label = 'Predicted Value')
plt.legend()
plt.xlabel('Values')
plt.ylabel('Gold price')
plt.title('True VS Predicted Values for Extra Tree')

plt.xlabel('Number of values')


plt.grid()
plt.show();
```



In [87]:

```
print ("The Accuracy of Extra Tree model is :",et_score*100,"%")
```

```
The Accuracy of Extra Tree model is : 98.85125775577237 %
```

In [ ]: