IMPORTING LIBRARIES:

In [2]:
```
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
from sklearn.tree import DecisionTreeRegressor
import xgboost as XGBRegressor
from sklearn.metrics import mean_squared_error
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_classif,f_regression
!pip install xgboost==1.0.1
import xgboost as xgb
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: xgboost==1.0.1 in c:\users\ruchi\anaconda3\lib\site-packages (1.0.1)
Requirement already satisfied: numpy in c:\users\ruchi\anaconda3\lib\site-packages (from xgboost==1.0.1) (1.24.3)
Requirement already satisfied: scipy in c:\users\ruchi\anaconda3\lib\site-packages (from xgboost==1.0.1) (1.10.1)
```

In [3]:
```
df=pd.read_csv("C:\\Users\\ruchi\\OneDrive\\Documents\\ML_dataset.csv")
```

In [4]:

```
pd.set_option("display.max_columns",100)
df.head()
```

Out[4]:

	*	Project Code	PQ #	PO / SO #	ASN/DN #	Country	Managed By	Fulfill Via	Vendor INCO Term	Shipment Mode	PQ First Sent to Client Date	PO Sent to Vendor Date	Scheduled Delivery Date	Delivered to Client Date	Delivery Recorded Date	Produ Grou
0	1	100-CI-T01	Pre-PQ Process	SCMS-4	ASN-8	Côte d'Ivoire	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	02-Jun-06	02-Jun-06	02-Jun-06	HRD
1	3	108-VN-T01	Pre-PQ Process	SCMS-13	ASN-85	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	14-Nov-06	14-Nov-06	14-Nov-06	AR
2	4	100-CI-T01	Pre-PQ Process	SCMS-20	ASN-14	Côte d'Ivoire	PMO - US	Direct Drop	FCA	Air	Pre-PQ Process	Date Not Captured	27-Aug-06	27-Aug-06	27-Aug-06	HRD
3	15	108-VN-T01	Pre-PQ Process	SCMS-78	ASN-50	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	01-Sep-06	01-Sep-06	01-Sep-06	AR
4	16	108-VN-T01	Pre-PQ Process	SCMS-81	ASN-55	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	11-Aug-06	11-Aug-06	11-Aug-06	AR

In [5]:

```
df.describe()
```

Out[5]:

	*	Unit of Measure (Per Pack)	Line Item Quantity	Line Item Value	Pack Price	Unit Price	Line Item Insurance (USD)
count	10324.000000	10324.000000	10324.000000	1.032400e+04	10324.000000	10324.000000	10037.000000
mean	51098.968229	77.990895	18332.534870	1.576506e+05	21.910241	0.611701	240.117626
std	31944.332496	76.579764	40035.302961	3.452921e+05	45.609223	3.275808	500.190568
min	1.000000	1.000000	1.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	12795.750000	30.000000	408.000000	4.314593e+03	4.120000	0.080000	6.510000
50%	57540.500000	60.000000	3000.000000	3.047147e+04	9.300000	0.160000	47.040000
75%	83648.250000	90.000000	17039.750000	1.664471e+05	23.592500	0.470000	252.400000
max	86823.000000	1000.000000	619999.000000	5.951990e+06	1345.640000	238.650000	7708.440000

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10324 entries, 0 to 10323
Data columns (total 33 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   *                             10324 non-null  int64
 1   Project Code                  10324 non-null  object
 2   PQ #                          10324 non-null  object
 3   PO / SO #                     10324 non-null  object
 4   ASN/DN #                      10324 non-null  object
 5   Country                       10324 non-null  object
 6   Managed By                    10324 non-null  object
 7   Fulfill Via                   10324 non-null  object
 8   Vendor INCO Term              10324 non-null  object
 9   Shipment Mode                 9964 non-null   object
 10  PQ First Sent to Client Date  10324 non-null  object
 11  PO Sent to Vendor Date        10324 non-null  object
 12  Scheduled Delivery Date       10324 non-null  object
 13  Delivered to Client Date      10324 non-null  object
 14  Delivery Recorded Date        10324 non-null  object
 15  Product Group                 10324 non-null  object
 16  Sub Classification            10324 non-null  object
 17  Vendor                        10324 non-null  object
 18  Item Description              10324 non-null  object
 19  Molecule/Test Type            10324 non-null  object
 20  Brand                         10324 non-null  object
 21  Dosage                        8588 non-null   object
 22  Dosage Form                   10324 non-null  object
 23  Unit of Measure (Per Pack)    10324 non-null  int64
 24  Line Item Quantity            10324 non-null  int64
 25  Line Item Value               10324 non-null  float64
 26  Pack Price                    10324 non-null  float64
 27  Unit Price                    10324 non-null  float64
 28  Manufacturing Site            10324 non-null  object
 29  First Line Designation        10324 non-null  object
 30  Weight (Kilograms)            10324 non-null  object
 31  Freight Cost (USD)            10324 non-null  object
 32  Line Item Insurance (USD)     10037 non-null  float64
dtypes: float64(4), int64(3), object(26)
memory usage: 2.6+ MB
```

DATA PRE-PROCESSING:

In [7]:
```
date_list=['Scheduled Delivery Date','Delivered to Client Date','Delivery Recorded Date']
same_date=[]
for i in range(len(date_list)-1):
    for j in df.index:
        if df[date_list[i]][j]!=df[date_list[i+1]][j]:
            same_date.append(j)
```

In [8]:
```
len(same_date)
```

Out[8]: 5911

In [9]:
```
date_list=['Scheduled Delivery Date','Delivered to Client Date','Delivery Recorded Date']
df['Date_deliver_client']=np.nan
df['Date_Delivery']=np.nan
df['Date_deliver_record']=np.nan

target=['Date_deliver_client','Date_Delivery','Date_deliver_record']
def date_input(target,present):
        for j in df.index:
            df[target][j]= df[present][j].split('-')[0]

for i in range(0,3):
    date_input(target[i],date_list[i])
```

In [10]:
```
df['month_deliver_client']=np.nan
df['month_Delivery']=np.nan
df['month_deliver_record']=np.nan
target=['month_deliver_client','month_Delivery','month_deliver_record']
def month_input(target,present):
        for j in df.index:
            df[target][j]= df[present][j].split('-')[1]
for i in range(0,3):
    month_input(target[i],date_list[i])
```

In [11]:

```
df['year_deliver_client']=np.nan
df['year_Delivery']=np.nan
df['year_deliver_record']=np.nan
target=['year_deliver_client','year_Delivery','year_deliver_record']
def year_input(target,present):
        for j in df.index:
            df[target][j]= df[present][j].split('-')[2]
for i in range(0,3):
    year_input(target[i],date_list[i])
```

In [12]:
```
df.head()
```

Out[12]:

	*	Project Code	PQ #	PO / SO #	ASN/DN #	Country	Managed By	Fulfill Via	Vendor INCO Term	Shipment Mode	PQ First Sent to Client Date	PO Sent to Vendor Date	Scheduled Delivery Date	Delivered to Client Date	Delivery Recorded Date	Produ Gro
0	1	100-CI-T01	Pre-PQ Process	SCMS-4	ASN-8	Côte d'Ivoire	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	02-Jun-06	02-Jun-06	02-Jun-06	HRD
1	3	108-VN-T01	Pre-PQ Process	SCMS-13	ASN-85	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	14-Nov-06	14-Nov-06	14-Nov-06	AF
2	4	100-CI-T01	Pre-PQ Process	SCMS-20	ASN-14	Côte d'Ivoire	PMO - US	Direct Drop	FCA	Air	Pre-PQ Process	Date Not Captured	27-Aug-06	27-Aug-06	27-Aug-06	HRD
3	15	108-VN-T01	Pre-PQ Process	SCMS-78	ASN-50	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	01-Sep-06	01-Sep-06	01-Sep-06	AF
4	16	108-VN-T01	Pre-PQ Process	SCMS-81	ASN-55	Vietnam	PMO - US	Direct Drop	EXW	Air	Pre-PQ Process	Date Not Captured	11-Aug-06	11-Aug-06	11-Aug-06	AF

In [13]:
```
df.drop(date_list,axis=1,inplace=True)
```

In [14]:

```
df.dtypes
```

Out[14]:

```
*                                int64
Project Code                    object
PQ #                            object
PO / SO #                       object
ASN/DN #                        object
Country                         object
Managed By                      object
Fulfill Via                     object
Vendor INCO Term                object
Shipment Mode                   object
PQ First Sent to Client Date    object
PO Sent to Vendor Date          object
Product Group                   object
Sub Classification              object
Vendor                          object
Item Description                object
Molecule/Test Type              object
Brand                           object
Dosage                          object
Dosage Form                     object
Unit of Measure (Per Pack)       int64
Line Item Quantity               int64
Line Item Value                float64
Pack Price                     float64
Unit Price                     float64
Manufacturing Site              object
First Line Designation          object
Weight (Kilograms)              object
Freight Cost (USD)              object
Line Item Insurance (USD)      float64
Date_deliver_client             object
Date_Delivery                   object
Date_deliver_record             object
month_deliver_client            object
month_Delivery                  object
month_deliver_record            object
year_deliver_client             object
year_Delivery                   object
year_deliver_record             object
dtype: object
```

In [15]:

```
wt_list=df['Weight (Kilograms)'].tolist()
wt_new={}
for i in range(len(wt_list)):
    try:
        pd.to_numeric(wt_list[i])

    except:
        wt_new[i]=wt_list[i]
len(wt_new)
df['wt_cat']=np.nan
```

In [16]:

```
for i in wt_new.keys():
    df['wt_cat'][i]=wt_new[i]
    df['Weight (Kilograms)'][i]=0
```

In [17]:

```
df['Weight (Kilograms)']=df['Weight (Kilograms)'].astype(int)
```

In [18]:

```
ft_list=df['Freight Cost (USD)'].tolist()
ft_new={}
for i in range(len(ft_list)):
    try:
        pd.to_numeric(ft_list[i])

    except:
        ft_new[i]=ft_list[i]
len(ft_new)
df['ft_cat']=np.nan
for i in ft_new.keys():
    df['ft_cat'][i]=ft_new[i]
    df['Freight Cost (USD)'][i]=0
df['Freight Cost (USD)']=df['Freight Cost (USD)'].astype(float)
```

In [19]:

```
df.dtypes
```

Out[19]:

*	int64
Project Code	object
PQ #	object
PO / SO #	object
ASN/DN #	object
Country	object
Managed By	object
Fulfill Via	object
Vendor INCO Term	object
Shipment Mode	object
PQ First Sent to Client Date	object
PO Sent to Vendor Date	object
Product Group	object
Sub Classification	object
Vendor	object
Item Description	object
Molecule/Test Type	object
Brand	object
Dosage	object
Dosage Form	object
Unit of Measure (Per Pack)	int64
Line Item Quantity	int64
Line Item Value	float64
Pack Price	float64
Unit Price	float64
Manufacturing Site	object
First Line Designation	object
Weight (Kilograms)	int32
Freight Cost (USD)	float64
Line Item Insurance (USD)	float64
Date_deliver_client	object
Date_Delivery	object
Date_deliver_record	object
month_deliver_client	object
month_Delivery	object
month_deliver_record	object
year_deliver_client	object
year_Delivery	object
year_deliver_record	object
wt_cat	object

```
ft_cat                          object
dtype: object
```

In [20]:

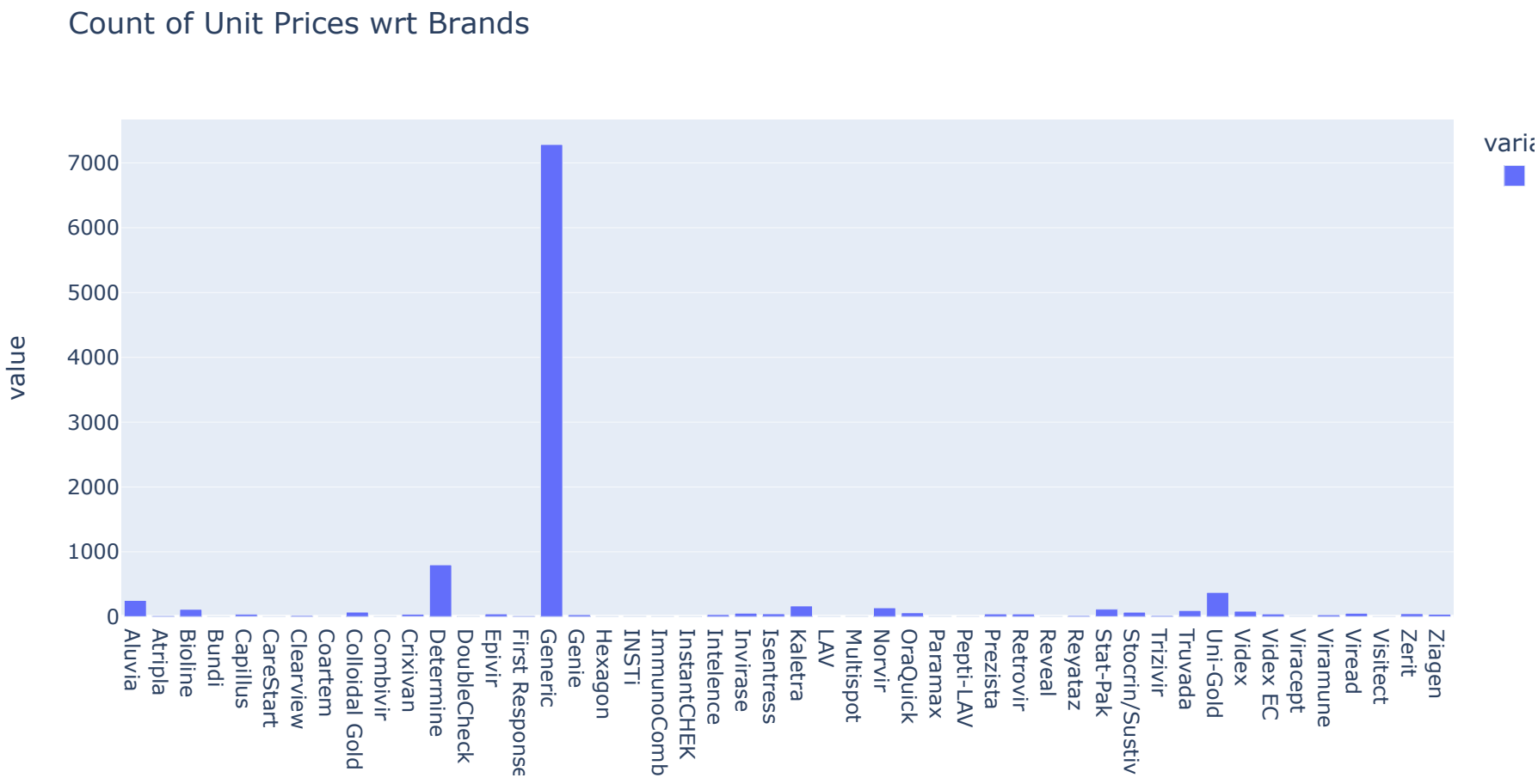
```python
df['Dosage Form'].fillna(-1,inplace=True)
df['Line Item Insurance (USD)'].fillna(0,inplace=True)
df.to_excel("C:final_sheet.xlsx")
```

DATA VISUALIZATION:

In [21]:

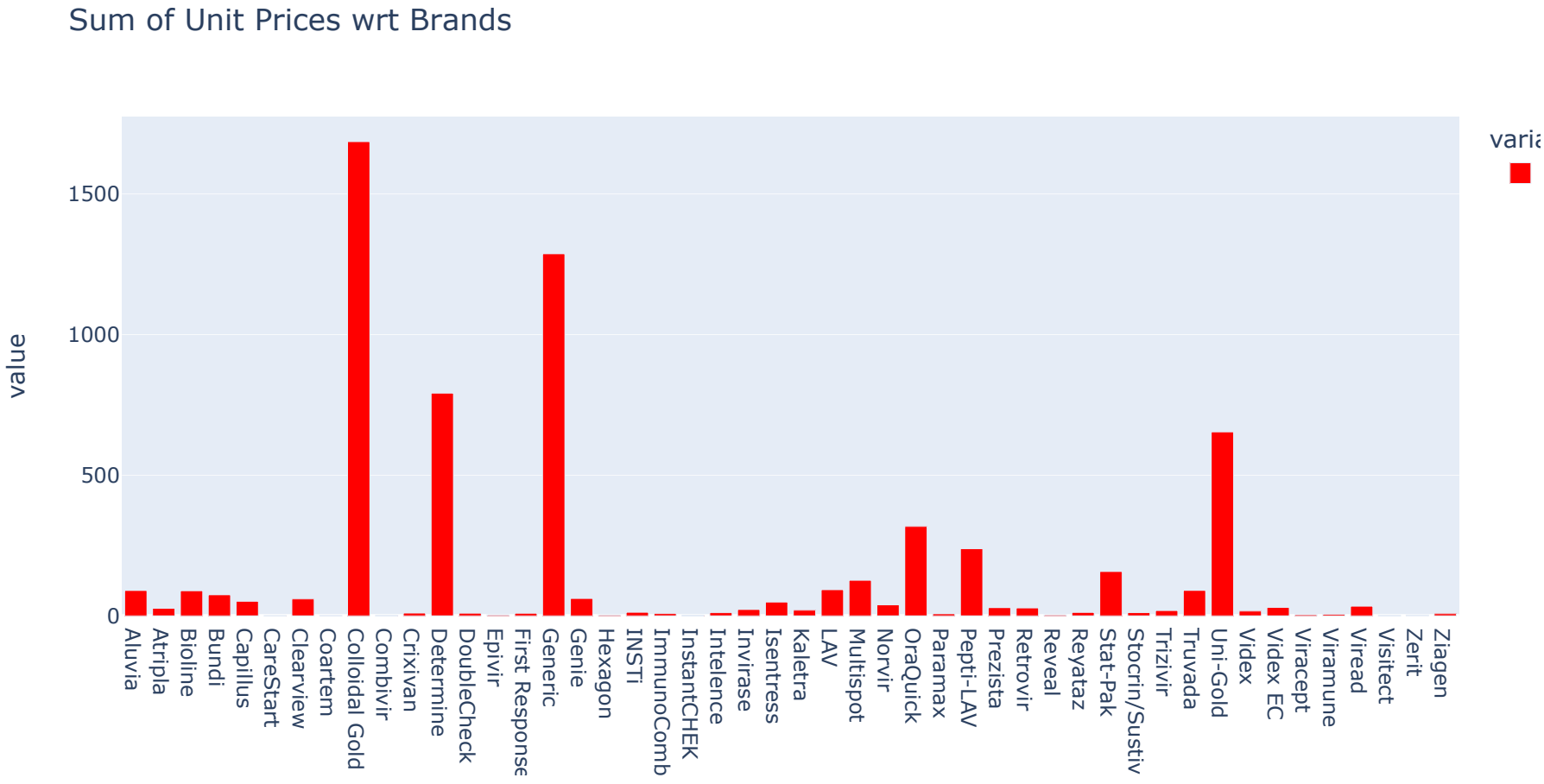
```
print("The total sum of unit price is ", df[df['Brand']=='Generic']['Unit Price'].sum())
px.bar(df.groupby('Brand')['Unit Price'].count(), title='Count of Unit Prices wrt Brands')
```

```
The total sum of unit price is  1287.3200000000002
```



In [22]:

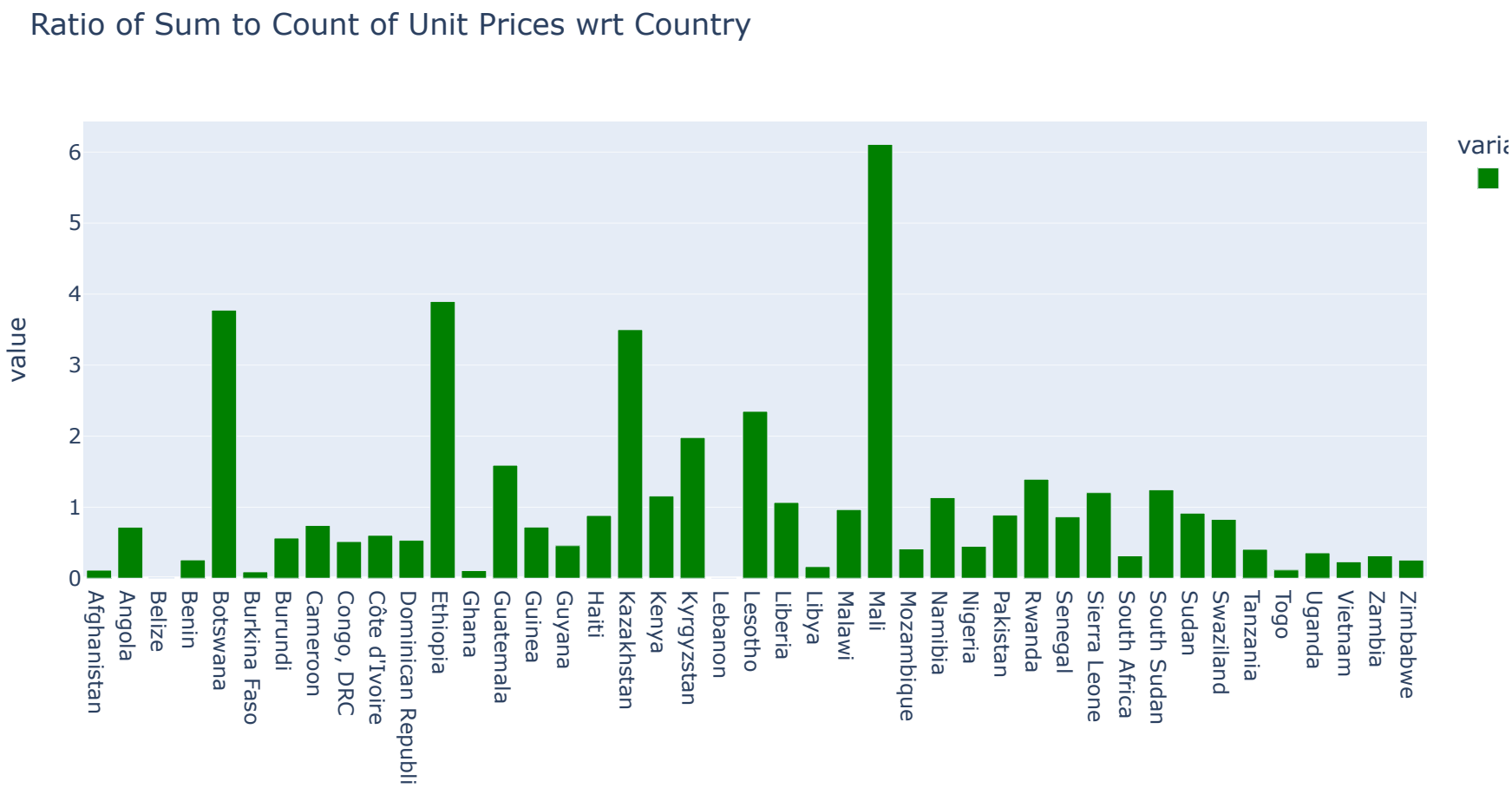
```
px.bar(df.groupby('Brand')['Unit Price'].sum(), title='Sum of Unit Prices wrt Brands',color_discrete_sequence =['r
```



In [23]:

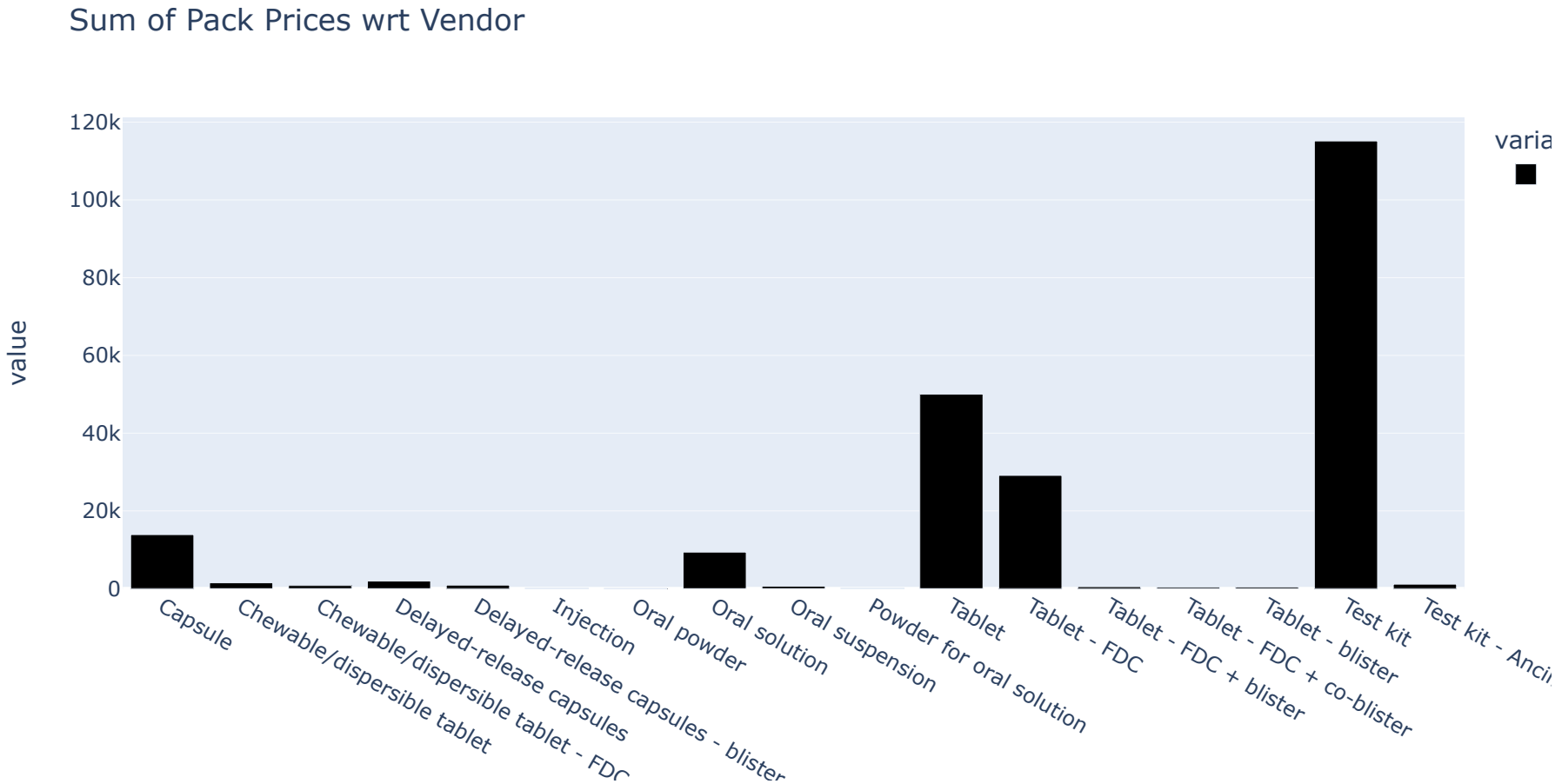
```
px.bar(df.groupby('Country')['Unit Price'].sum()/df.groupby('Country')['Unit Price'].count(), title='Ratio of Sum

```



In [24]:

```
px.bar(df.groupby('Dosage Form')['Pack Price'].sum(), title='Sum of Pack Prices wrt Vendor',color_discrete_sequenc
```



INITIALIZATION:

In [25]:
```
X=df.drop(['Pack Price','Unit Price'],axis=1)
```

In [26]:
```
y=df['Pack Price']
Y=df['Unit Price']
```

In [27]:
```
cat_col=[]
num_col=[]
for i in X.columns:
    if X[i].dtype=='O':
        cat_col.append(i)
    else:
        num_col.append(i)
categorical_transformer = OneHotEncoder(handle_unknown='ignore')
```

In [28]:
```
import pickle
pickle.dump(categorical_transformer, open('cattrans.pkl','wb'))
```

In [29]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline

numeric_transformer = Pipeline(steps=[('scaler',MinMaxScaler()) ])
pickle.dump(numeric_transformer, open('numerictrans.pkl','wb'))
preprocessing=ColumnTransformer(remainder='passthrough', transformers=[('cat', categorical_transformer, cat_col),
pickle.dump(preprocessing, open('prepro.pkl','wb'))
```

#FOR UNITPRICE:

In [30]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,random_state=0)
```

In [31]:

```
import pandas as pd

date_str = '12-05-10'
date_obj = pd.to_datetime(date_str, format='%y-%m-%d')
timestamp = date_obj.timestamp()  # Converts to a float representing the timestamp
# Example usage
print(timestamp)

final_list=[]
model=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor(),xgb.XGBRegressor()]
result={}
for j in  [5, 10, 15, 20, '25', '30','35','39']:
        new=[]
        for i in model:

            runpipe = Pipeline(steps=[('preprocessor', preprocessing), ('skb',SelectKBest(score_func=f_regression,
                                ('model', i)])
            runpipe.fit(X_train,Y_train)
            pred=runpipe.predict(X_test)
            rmse = np.sqrt(mean_squared_error(Y_test, pred))
            new.append(runpipe.score(X_test, Y_test))
        final_list.append(new)
```

```
1336608000.0
```

In [32]:

```
pd.DataFrame(final_list,index=['5', '10', '15', '20', '25', '30','35','39'], columns=['Linear Regression','Decisio
```

Out[32]:

	Linear Regression	Decision Tree Regressor	Random Forest Regressor	XGBRegressor()
5	-0.002967	-0.002967	-0.002678	-0.002967
10	0.731158	0.731158	0.731148	0.731158
15	0.816197	0.816197	0.815141	0.816198
20	0.820665	0.824756	0.825527	0.824757
25	0.820532	0.824191	0.823992	0.824193
30	0.821485	0.825144	0.824633	0.825166
35	0.891285	0.895023	0.895634	0.895026
39	0.890749	0.852023	0.852094	0.851924

#FOR PACK_PRICE:

In [33]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
                                                    random_state=0)
final_list=[]
model=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor(),xgb.XGBRegressor()]
result={}
for j in [5, 10, '15', '20', '25', '30','35','39']:
        new=[]
        for i in model:

            runpipe = Pipeline(steps=[('preprocessor', preprocessing), ('skb',SelectKBest(score_func=f_regression,
                              ('model', i)])
            runpipe.fit(X_train,y_train)
            pred=runpipe.predict(X_test)
            rmse = np.sqrt(mean_squared_error(y_test, pred))
            new.append(runpipe.score(X_test, y_test))
        final_list.append(new)
pd.DataFrame(final_list,index=['5', '10', '15', '20', '25', '30','35','39'], columns=['Linear Regression','Decisio
```

Out[33]:

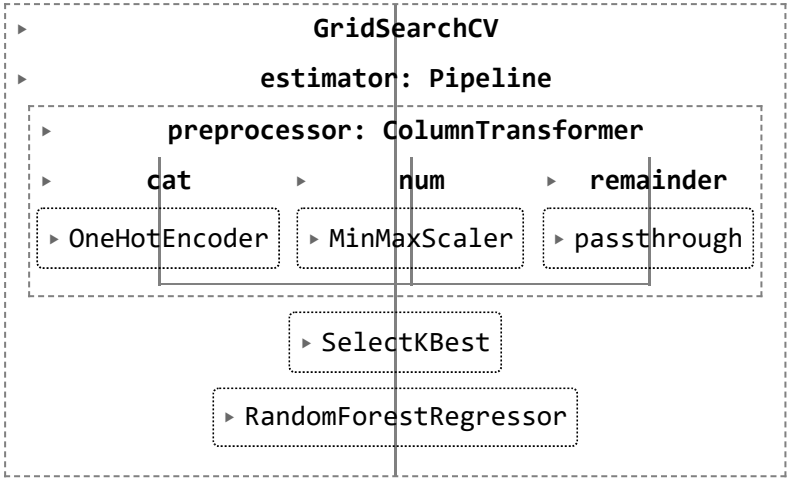
	Linear Regression	Decision Tree Regressor	Random Forest Regressor	XGBRegressor()
5	-0.002967	-0.002967	-0.003079	-0.002967
10	0.731158	0.731158	0.731174	0.731158
15	0.816197	0.816197	0.816519	0.816198
20	0.820665	0.824756	0.824899	0.824757
25	0.820532	0.824191	0.825693	0.824193
30	0.821485	0.825144	0.825129	0.825166
35	0.891285	0.895023	0.894615	0.895026
39	0.890749	0.852023	0.853343	0.851924

HYPERTUNING OF RANDOM FOREST

In [34]:

```
pipeline = Pipeline(steps=[('preprocessor', preprocessing), ('skb',SelectKBest(score_func=f_regression, k = int(j
                                ('model', RandomForestRegressor())])
parameters={'model__max_depth':[50,None],
    'model__min_samples_split':[2,5],
    'model__min_samples_leaf':[1,2]}
pickle.dump(preprocessing, open('prepro.pkl','wb'))
CV = GridSearchCV(pipeline, parameters, scoring = 'neg_mean_absolute_error', n_jobs= 1)
CV.fit(X_train, y_train)
```

Out[34]:



In [35]:

```
CV.best_params_
```

Out[35]:

```
{'model__max_depth': 50,
 'model__min_samples_leaf': 1,
 'model__min_samples_split': 2}
```

In [36]:

```
y_pred = CV.predict(X_test)
r2_score(y_test,y_pred)

```

Out[36]:

```
0.8545084237273932
```

In [ ]:

```
1
```