PROFIT PREDICTION OF USA SUPERSTORE CUSTOMERS

Explanatory Data Analysis

Feature Engineering

ML Algorithms Training

Metadata

Row ID => Unique ID for each row.
Order ID => Unique Order ID for each Customer.
Order Date => Order Date of the product.
Ship Date => Shipping Date of the Product.
Ship Mode=> Shipping Mode specified by the Customer.
Customer ID => Unique ID to identify each Customer.
Customer Name => Name of the Customer.
Segment => The segment where the Customer belongs.
Country => Country of residence of the Customer.
City => City of residence of of the Customer.
State => State of residence of the Customer.
Postal Code => Postal Code of every Customer.
Region => Region where the Customer belong.
Product ID => Unique ID of the Product.
Category => Category of the product ordered.
Sub-Category => Sub-Category of the product ordered.
Product Name => Name of the Product.
Sales => Sales of the Product.
Quantity => Quantity of the Product.
Discount => Discount provided.
Profit => Profit/Loss incurred.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

```
warnings.filterwarnings('ignore')
sns.set_theme(style='whitegrid')
```

In [2]:
```
df = pd.read_csv('Superstore.csv', encoding='unicode_escape')
#df = pd.read_csv('Superstore.csv', engine='python')
df.head(2)
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson

2 rows × 21 columns

In [3]:
```
df.tail(2)
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
9992	9993	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9993	9994	CA-2017-119914	5/4/2017	5/9/2017	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westmin

2 rows × 21 columns

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

Out[4]:
```
(9994, 21)
```

PART 1

Explanatory Data Analysis

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Row ID         9994 non-null   int64 
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
```

```
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.0+ MB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

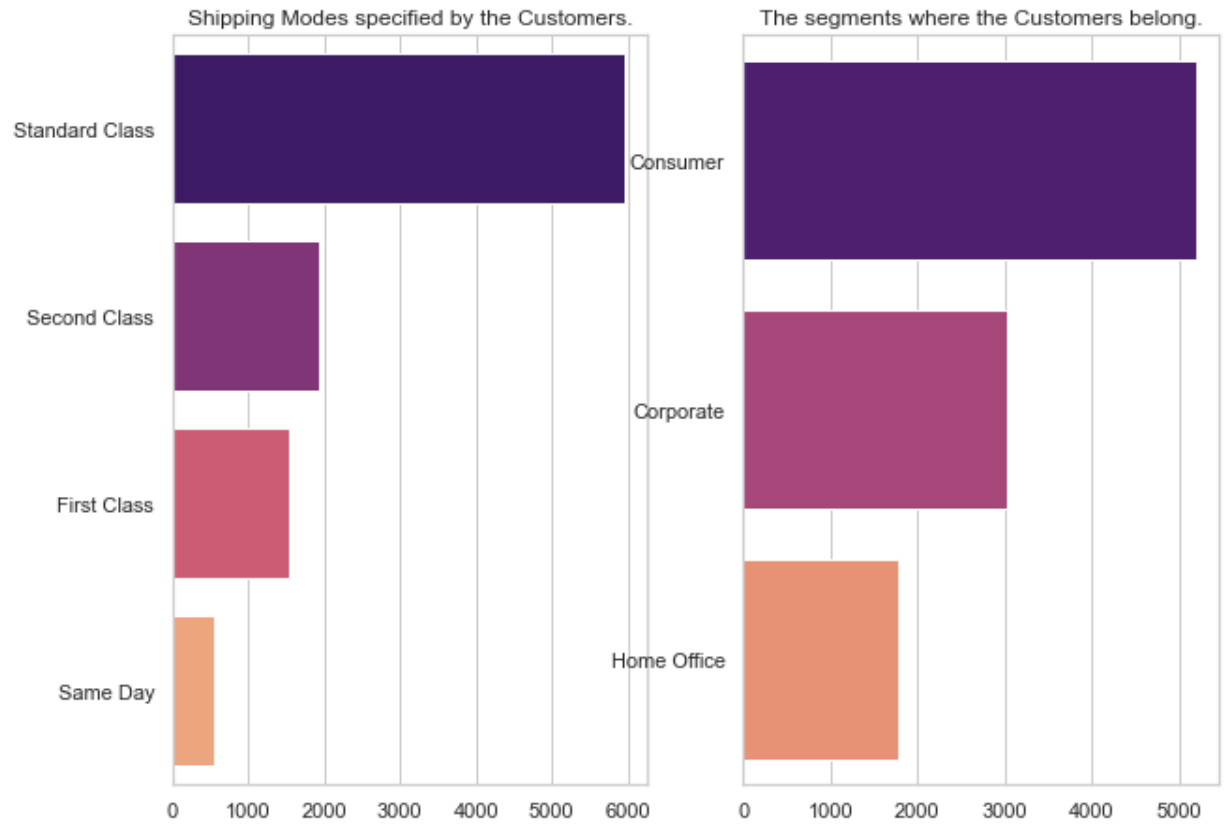
In [7]:
```
df.describe()
```

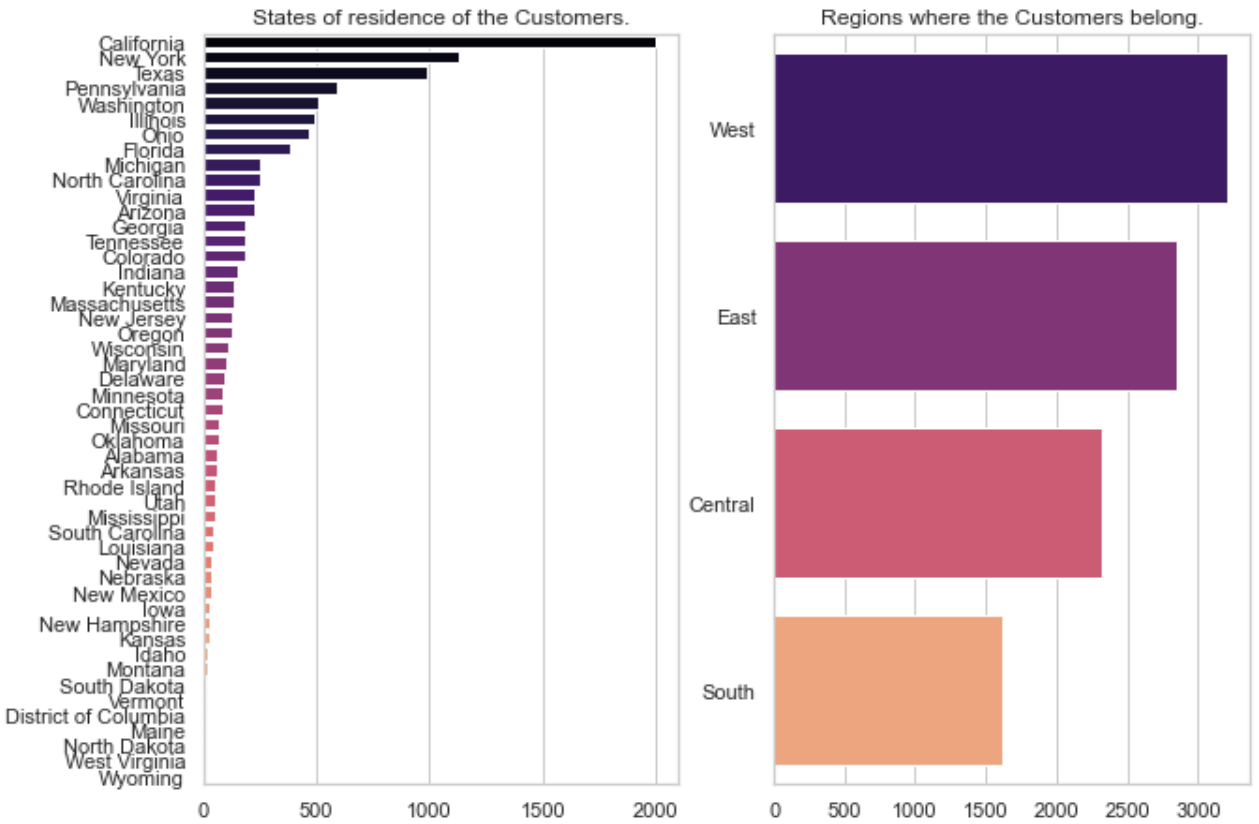
Out[7]:

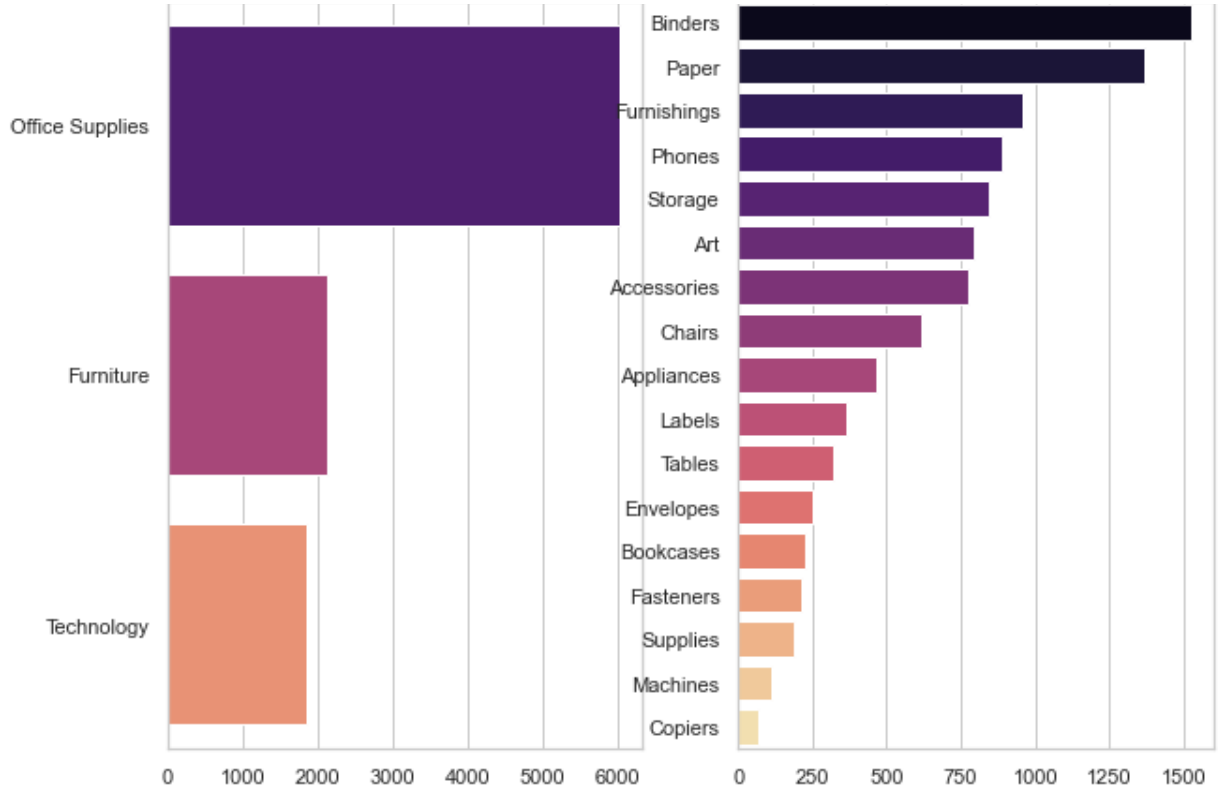
	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

In [8]:
```
categories = ['Ship Mode','Segment','State','Region', 'Category','Sub-Category']
indexrange = [1,2,3,4,5,6]
cat_class = []
y = []
x = []
title = ['Shipping Modes specified by the Customers.','The segments where the Cu
         'States of residence of the Customers.','Regions where the Customers be
         'Categories of the products ordered.','Sub-Categories of the products o
for i in range(6):
    i = df[categories[i]].value_counts()
    cat_class.append(i)
plt.figure(figsize = (10,25))
for i in range(6):
    y.append(cat_class[i].index)
    x.append(cat_class[i].values)
```

```
    plt.subplot(3,2,indexrange[i])
    sns.barplot(x[i], y[i], palette='magma')
    plt.title(title[i])
```





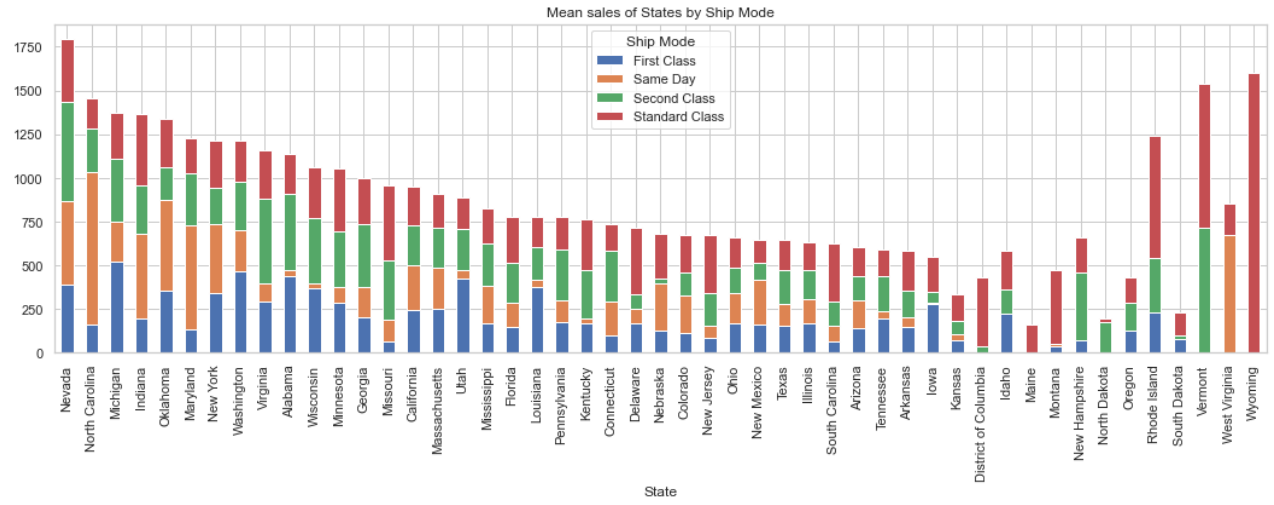


In [9]:
```
Sales_shipdf = df.groupby(['State','Ship Mode']).aggregate({'Sales':'mean'}).res
Sales_shipdf = Sales_shipdf.pivot(index='State', columns='Ship Mode', values='Sa
Sales_shipdf['total'] = Sales_shipdf['First Class']+Sales_shipdf['Same Day']+Sal

Sales_shipdf = Sales_shipdf.sort_values(by = 'total', ascending = False)
Sales_shipdf.drop(columns ='total', inplace=True)
Sales_shipdf.head()
```

Out[9]:

Ship Mode	State	First Class	Same Day	Second Class	Standard Class
26	Nevada	392.239600	475.944000	567.149667	356.484000
31	North Carolina	159.149263	875.506929	248.553333	173.284871
20	Michigan	520.668857	230.780222	356.993813	262.177506
12	Indiana	194.193125	483.973333	276.516857	413.876421
34	Oklahoma	356.334000	519.794286	184.144286	276.045745

In [10]:
```
Sales_shipdf.set_index('State').plot(kind = 'bar', figsize = (18,5), stacked = T
plt.title('Mean sales of States by Ship Mode')
plt.show()
```

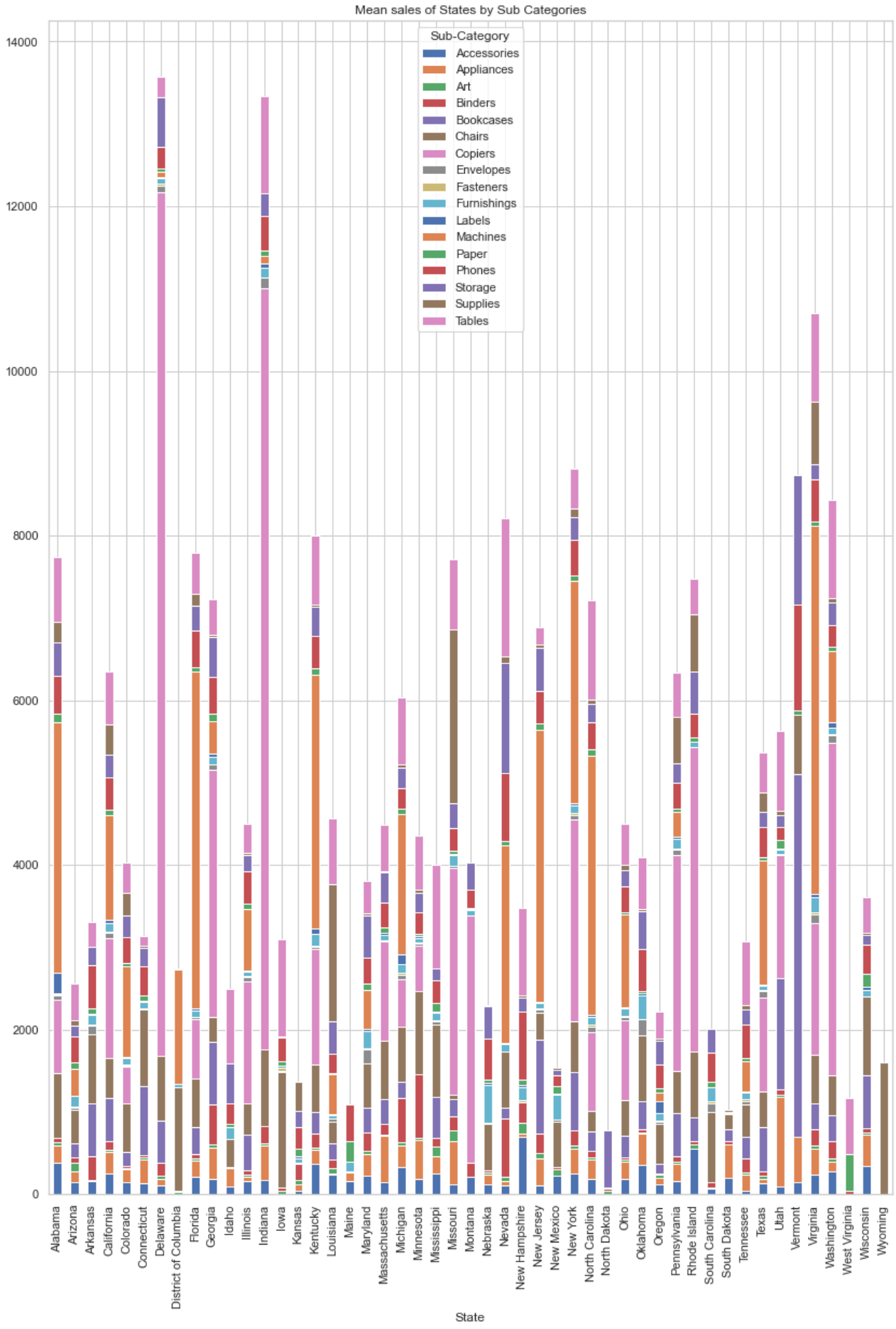


In [11]:
```
sub_df = df.groupby(['State','Sub-Category']).aggregate({'Sales':'mean'}).reset_
sub_df = sub_df.pivot(index = 'State', columns = 'Sub-Category', values = 'Sales
sub_df['total'] = sub_df[['Accessories','Appliances','Art','Binders','Bookcases'
sub_df.head()
```

Out[11]:

Sub-Category	State	Accessories	Appliances	Art	Binders	Bookcases	Chairs	Cop
0	Alabama	387.138333	208.160000	43.030000	46.682000	NaN	783.108000	899.970
1	Arizona	154.370909	129.072000	99.225143	62.438657	173.071000	406.192500	N
2	Arkansas	162.585455	NaN	13.050000	288.983000	638.820000	836.640000	N
3	California	253.435442	257.190638	33.307030	103.163652	529.971567	485.225908	1444.677
4	Colorado	152.532800	151.921778	19.147636	18.177429	175.947000	588.915429	439.992

In [12]:
```
sub_df = sub_df.sort_values(by = 'total', ascending = False)
sub_df.drop(columns ='total', inplace=True)
sub_df.set_index('State').plot(kind = 'bar', figsize = (14,20), stacked = True)
plt.title("Mean sales of States by Sub Categories")
plt.show()
```



In [13]:
```
pf_seg_df = df.groupby(['State','Segment']).aggregate({'Profit':'mean'}).reset_i
```

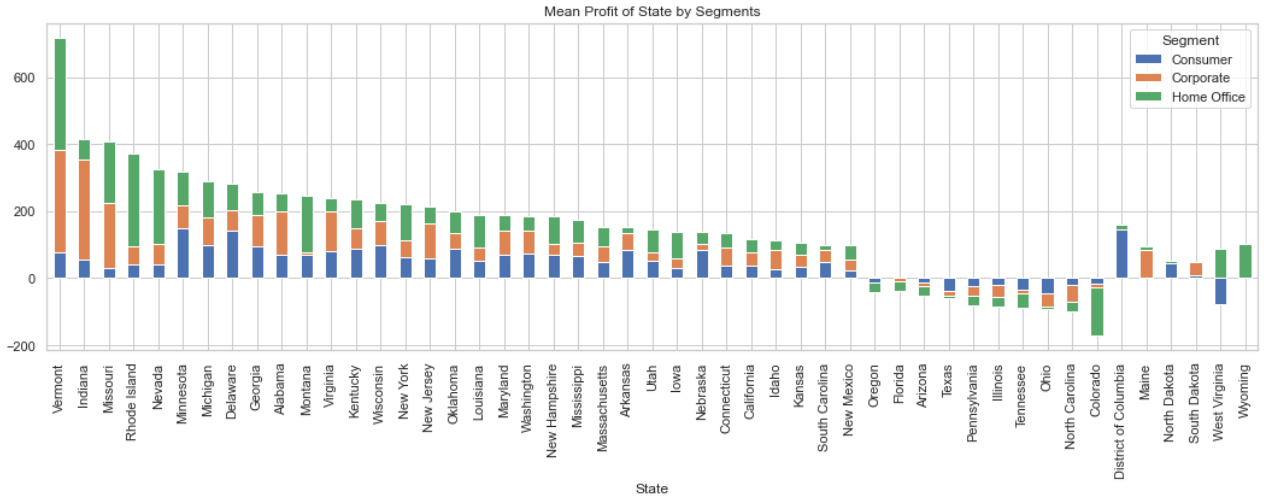
```
pf_seg_df = pf_seg_df.pivot(index='State', columns='Segment', values='Profit').r
pf_seg_df['total'] = pf_seg_df['Consumer']+pf_seg_df['Corporate']+pf_seg_df['Hom

pf_seg_df = pf_seg_df.sort_values(by = 'total', ascending = False)
pf_seg_df.drop(columns ='total', inplace=True)
pf_seg_df.head()
```

Out[13]:

Segment	State	Consumer	Corporate	Home Office
43	Vermont	75.023200	306.645460	336.635000
12	Indiana	55.153593	299.846849	58.942949
23	Missouri	30.475097	193.666453	182.550769
37	Rhode Island	41.127433	54.708805	275.991645
26	Nevada	40.759328	59.706515	225.864162

In [14]:

```
pf_seg_df.set_index('State').plot(kind = 'bar', figsize = (18,5), stacked = True
plt.title("Mean Profit of State by Segments")
plt.show()
```



In [15]:

```
q_r_df = df.groupby(['State','Ship Mode']).aggregate({'Quantity':'sum'}).reset_i
q_r_df = q_r_df.pivot(index='State', columns='Ship Mode', values='Quantity').res
q_r_df['total'] = q_r_df['First Class']+q_r_df['Same Day']+q_r_df['Second Class'


q_r_df = q_r_df.sort_values(by = 'total', ascending = False)
q_r_df.drop(columns ='total', inplace=True)
q_r_df.head()
```

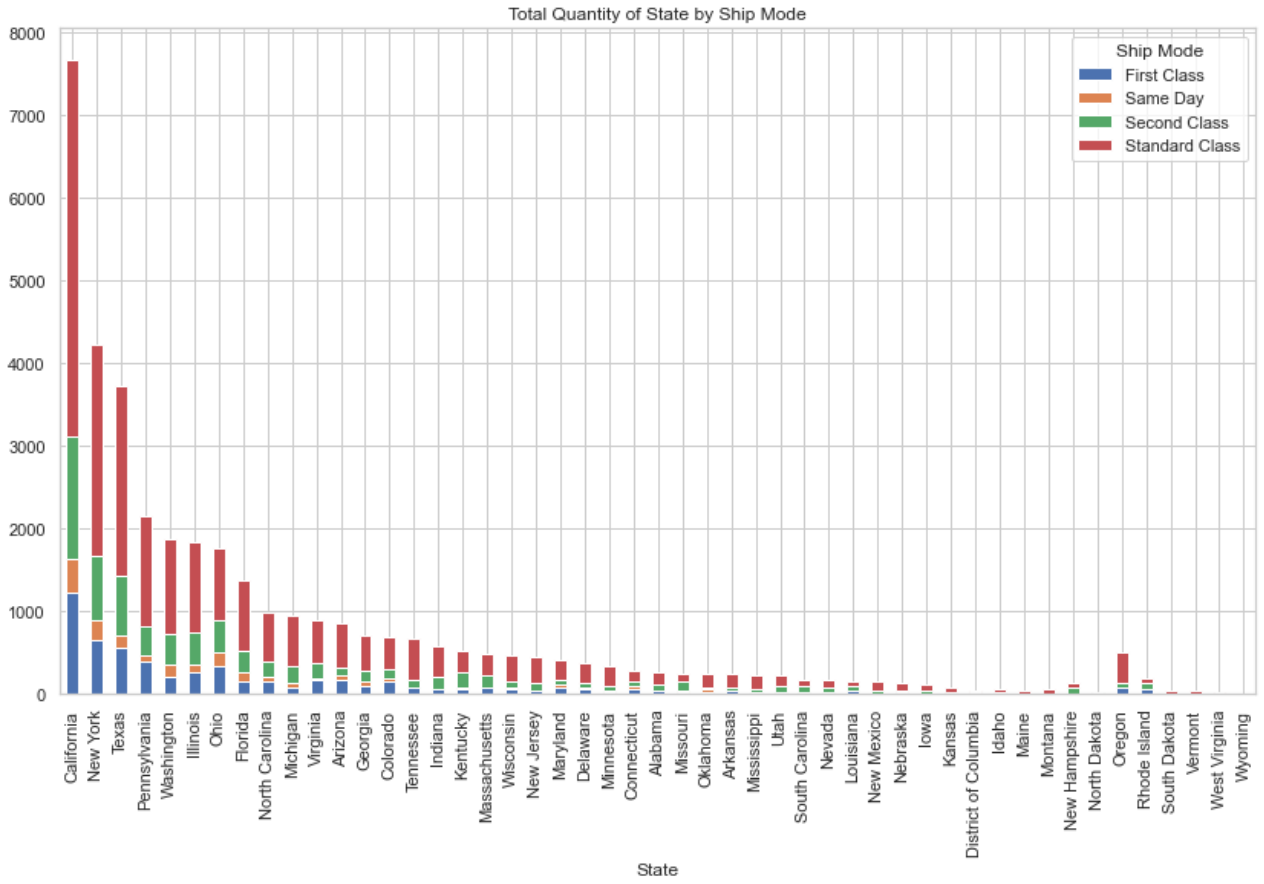
Out[15]:

Ship Mode	State	First Class	Same Day	Second Class	Standard Class
3	California	1233.0	404.0	1475.0	4555.0
30	New York	651.0	250.0	768.0	2555.0
41	Texas	560.0	158.0	718.0	2288.0
36	Pennsylvania	399.0	69.0	353.0	1332.0
45	Washington	211.0	152.0	372.0	1148.0

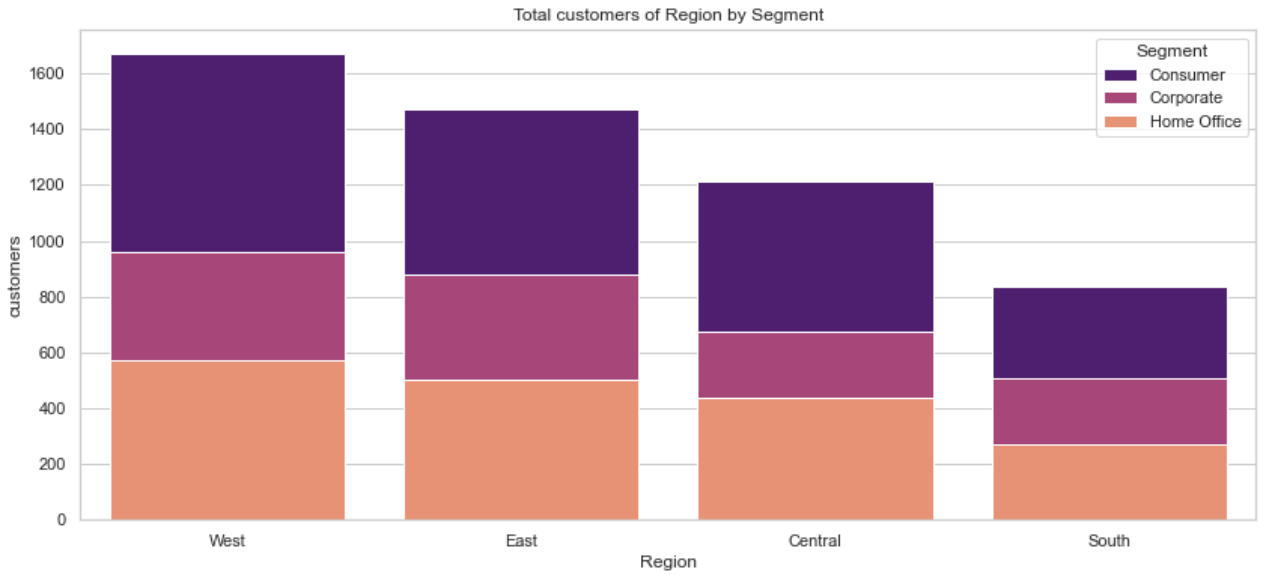
In [16]:

```
q_r_df.set_index('State').plot(kind = 'bar', figsize = (14,8), stacked = True)
```

```
plt.title("Total Quantity of State by Ship Mode")
plt.show()
```



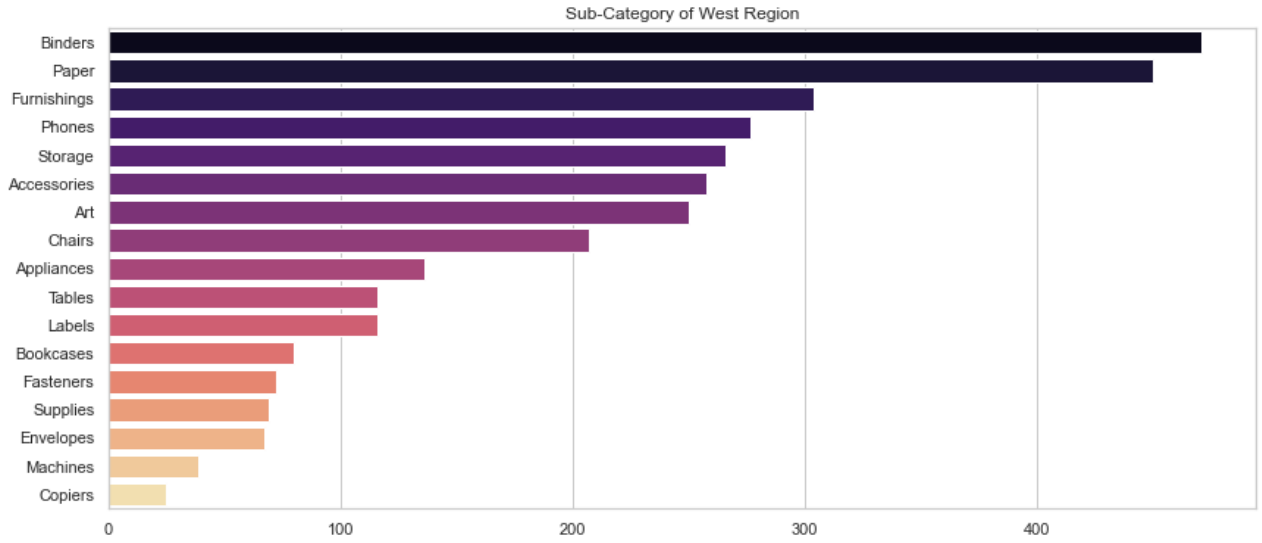
In [17]:
```
plt.figure(figsize=(14,6))
region_df = df.groupby(['Region', 'Segment']).size().reset_index().rename(column
region_df.pivot(columns = 'Region', index = 'Segment', values = 'customers')
sns.barplot(x='Region', y='customers', data=region_df, palette='magma', hue='Seg
plt.title("Total customers of Region by Segment")
plt.show()
```



Analysis of most customers in some categories

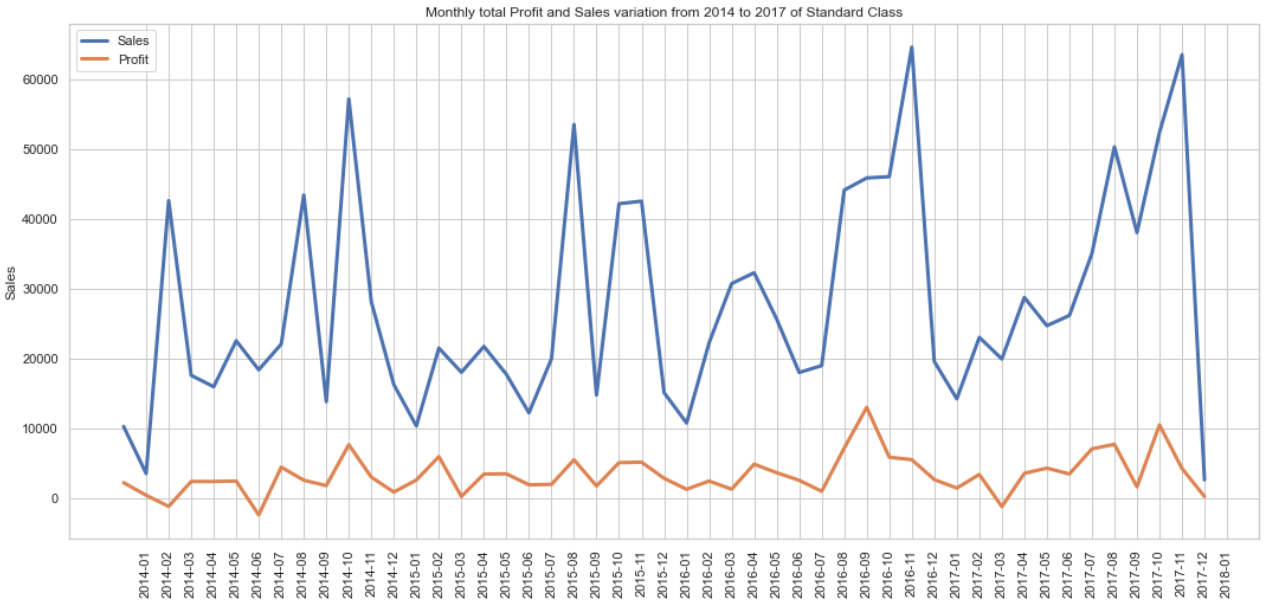
The West Region which was the having most customers.

In [18]:
```
west_df = df.loc[df['Region']=='West']

west_df['Sub-Category'].value_counts().reset_index()

plt.figure(figsize=(14,6))
sns.barplot(x=west_df['Sub-Category'].value_counts().values, y=west_df['Sub-Cate
plt.title("Sub-Category of West Region")
plt.show()
```



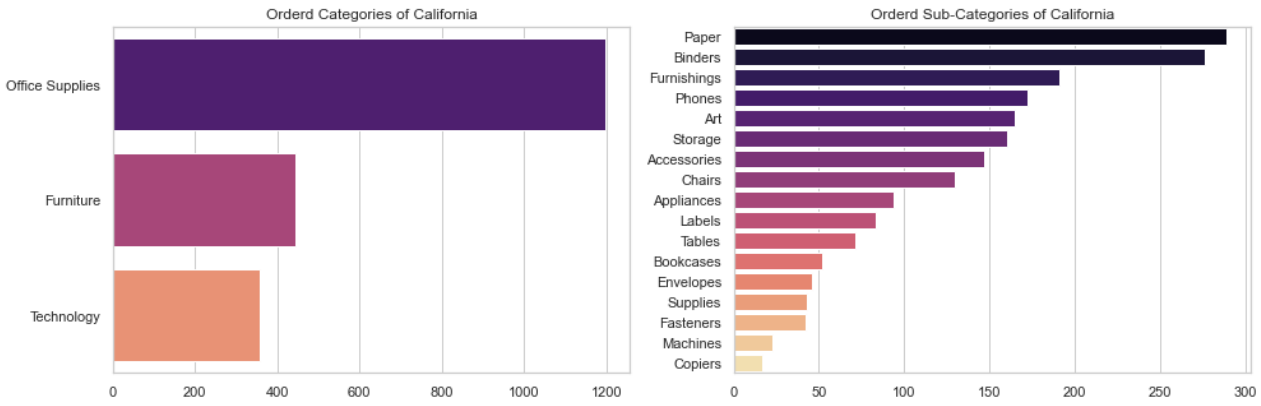
The Standard Class in Ship-Mode which was the having most customers.

In [19]:
```
stndclz_df = df.loc[df['Ship Mode'] == 'Standard Class']
stndclz_df = stndclz_df[['Ship Date', 'Sales', 'Profit']].sort_values(by='Ship D
stndclz_df['monthyr'] = pd.to_datetime(stndclz_df['Ship Date']).dt.to_period('M'
stndclz_df = stndclz_df.groupby('monthyr').agg({'Sales':'sum', 'Profit':'sum'}).

plt.figure(figsize=(18,8))
sns.lineplot(x = stndclz_df.index, y = 'Sales', data = stndclz_df, label = 'Sale
sns.lineplot(x = stndclz_df.index, y = 'Profit', data = stndclz_df, label = 'Pro

labels = stndclz_df['monthyr'].values
plt.xticks(range(1,stndclz_df.shape[0]+1), labels=labels)
plt.xticks(rotation=90)
plt.title('Monthly total Profit and Sales variation from 2014 to 2017 of Standar
plt.show()
```



California State which was the having most customers.

In [20]:
```
california_df = df.loc[df['State'] == 'California']

plt.figure(figsize=(16,5))
plt.subplot(1,2,2)
sns.barplot(california_df['Sub-Category'].value_counts().values, california_df['
plt.title('Orderd Sub-Categories of California')

plt.subplot(1,2,1)
sns.barplot(california_df['Category'].value_counts().values, california_df['Cate
plt.title('Orderd Categories of California')

plt.show()
```

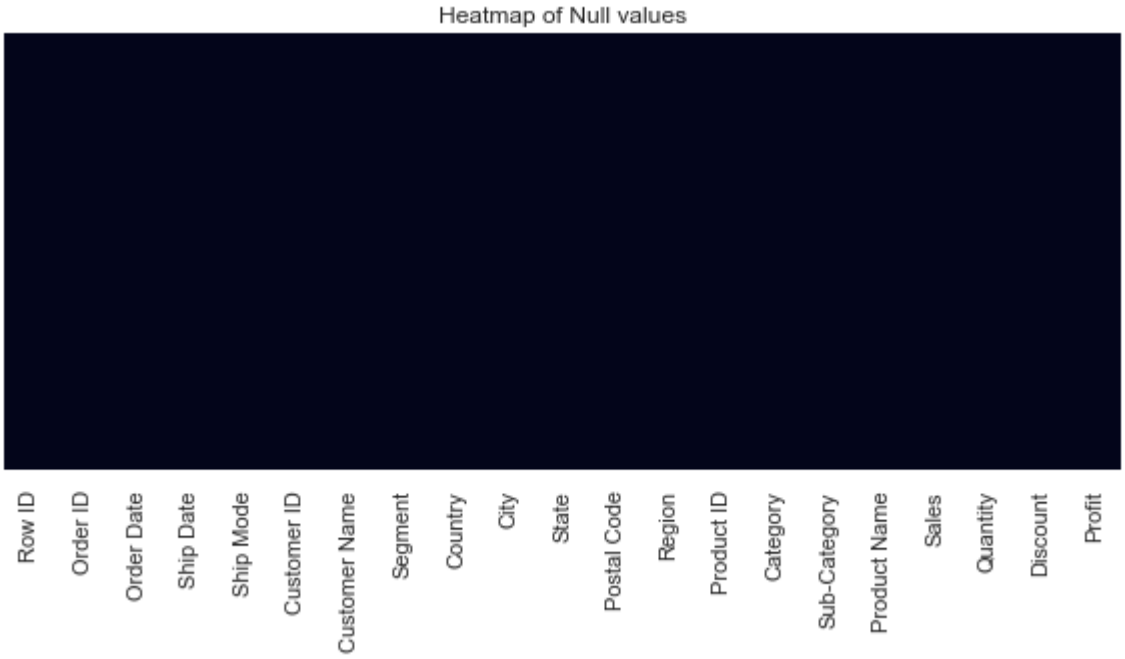


PART 2

Feature Engineering

In [21]:
```
df_drop = df.drop(['Row ID', 'Order ID','City', 'Order Date', 'Ship Date', 'Cust
```

In [22]:
```
plt.figure(figsize=(10,4))
sns.heatmap(df.isnull(), yticklabels=False, cbar = False)
plt.title('Heatmap of Null values')
plt.show()
```



In [23]:
```
df_drop.head()
```

Out[23]:

	Ship Mode	Segment	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Second Class	Consumer	Kentucky	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	Second Class	Consumer	Kentucky	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	Second Class	Corporate	California	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	Standard Class	Consumer	Florida	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	Standard Class	Consumer	Florida	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

In [24]:
```
scaler = StandardScaler()

scalecol = ['Sales', 'Quantity', 'Discount', 'Profit']
for i in scalecol:
    df_drop[i] = scaler.fit_transform(df_drop[[i]])
```

In [25]:
```
lablecol = df_drop[['Ship Mode', 'Segment', 'State', 'Region', 'Category','Sub-C
dummi = pd.get_dummies(lablecol,drop_first=True)
```

In [26]:
```
df_drop = pd.concat([df_drop,dummi], axis=1)
df_drop = df_drop.drop(['Ship Mode', 'Segment', 'State', 'Region', 'Category','S
df_drop.head()
```

Out[26]:

	Sales	Quantity	Discount	Profit	Ship Mode_Same Day	Ship Mode_Second Class	Ship Mode_Standard Class	Segment_Corp
0	0.051510	-0.804303	-0.756643	0.056593	0	1	0	
1	0.805633	-0.354865	-0.756643	0.815054	0	1	0	

	Sales	Quantity	Discount	Profit	Ship Mode_Same Day	Ship Mode_Second Class	Ship Mode_Standard Class	Segment_Corp
2	-0.345368	-0.804303	-0.756643	-0.093002	0	1	0	
3	1.167688	0.544012	1.423149	-1.757484	0	0	1	
4	-0.332935	-0.804303	0.212153	-0.111593	0	0	1	

5 rows × 78 columns

PART 3

Models Training

In [27]:
```
y = df_drop['Profit']
x = df_drop.drop(['Profit'], axis=1)
x.shape, y.shape
```

Out[27]:
```
((9994, 77), (9994,))
```

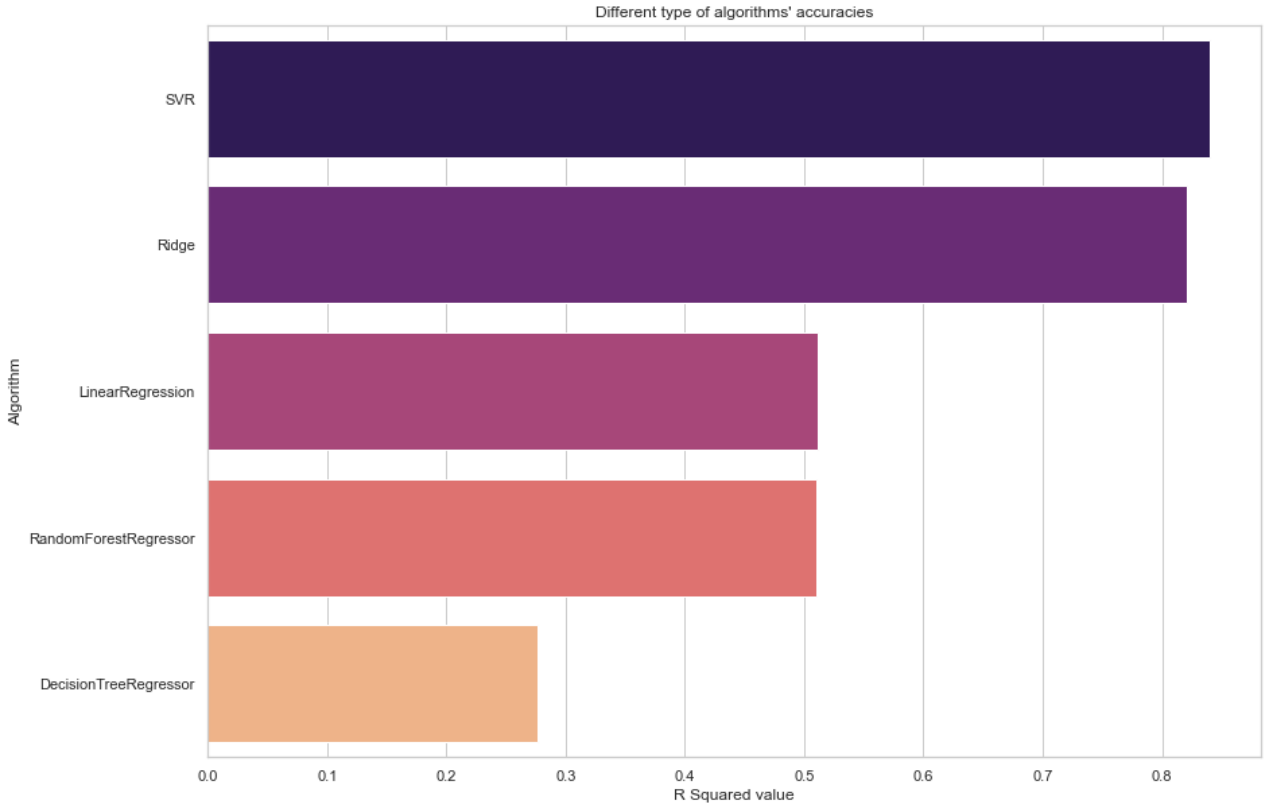
In [28]:
```
x_train,x_test, y_train,y_test = train_test_split(x,y, test_size=0.2,random_stat
```

In [29]:
```
models = ['SVR','Ridge', 'LinearRegression', 'RandomForestRegressor', 'DecisionT

svr_model = SVR(kernel='rbf').fit(x_train,y_train)
ridge_model = Ridge().fit(x_train,y_train)
lr_model = LinearRegression().fit(x_train,y_train)
rf_model = RandomForestRegressor(n_estimators=10, random_state=0).fit(x_train,y_
dt_model = DecisionTreeRegressor(random_state=0).fit(x_train,y_train)

svr_pred = svr_model.predict(x_test)
ridge_pred = ridge_model.predict(x_test)
lr_pred = lr_model.predict(x_test)
rf_pred = rf_model.predict(x_test)
dt_pred = dt_model.predict(x_test)

R2_values = [r2_score(y_test,svr_pred),r2_score(y_test,ridge_pred),
             r2_score(y_test,lr_pred),r2_score(y_test,rf_pred),r2_score(y_test,d
```

In [30]:
```
R2_values = sorted(R2_values, reverse=True)
R2_values[0]
```

Out[30]:
```
0.8405019729439146
```

In [31]:
```
plt.figure(figsize=(14,10))
sns.barplot(y = models, x = R2_values, palette = 'magma')
plt.title("Different type of algorithms' accuracies")
plt.xlabel('R Squared value')
plt.ylabel('Algorithm')
plt.show()
```



Parameter Tuning

Out of this five algorithms, there are 2 algorithms which fitting the accuracies more than 80%. So out of them the Random Forest Regression was the best algorithm for the particular dataset. Let's do parameter tuning for it to see whether the acuuracy can be increased.

Hyperparameter tuning with RandomizedSerachCV

In [32]:
```
%%time

from sklearn.model_selection import RandomizedSearchCV

#Different RandomForestRegressor hyperparameters
rf_grid = {'n_estimators': np.arange(10,100,10),
           'max_depth': [None,3,5,10],
           'min_samples_split':np.arange(2,20,2),
           'min_samples_leaf':np.arange(1,20,2),
           'max_features':[0.5,1,'sqrt','auto'],
           'max_samples':[3000],
           'criterion':['mae','mse']}

#Instantiate RandomizedSerachcCV model
rs_model = RandomizedSearchCV(RandomForestRegressor(n_jobs = -1,
                                                    random_state=3),
                              param_distributions=rf_grid,
                              n_iter=3,
                              cv = 5,
                              verbose=True)

#fit the RandomizedSearchCV model
rs_model.fit(x, y)
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Wall time: 31.7 s
```

Out[32]:
```
RandomizedSearchCV(cv=5,
                   estimator=RandomForestRegressor(n_jobs=-1, random_state=3),
                   n_iter=3,
                   param_distributions={'criterion': ['mae', 'mse'],
                                        'max_depth': [None, 3, 5, 10],
                                        'max_features': [0.5, 1, 'sqrt',
                                                         'auto'],
                                        'max_samples': [3000],
                                        'min_samples_leaf': array([ 1,  3,  5,
7,  9, 11, 13, 15, 17, 19]),
                                        'min_samples_split': array([ 2,  4,  6,
8, 10, 12, 14, 16, 18]),
                                        'n_estimators': array([10, 20, 30, 40, 5
0, 60, 70, 80, 90])},
                   verbose=True)
```

In [33]:
```
#Find the best model hyperparameters
rs_model.best_params_
```

Out[33]:
```
{'n_estimators': 60,
 'min_samples_split': 8,
 'min_samples_leaf': 5,
 'max_samples': 3000,
 'max_features': 'sqrt',
 'max_depth': None,
 'criterion': 'mse'}
```

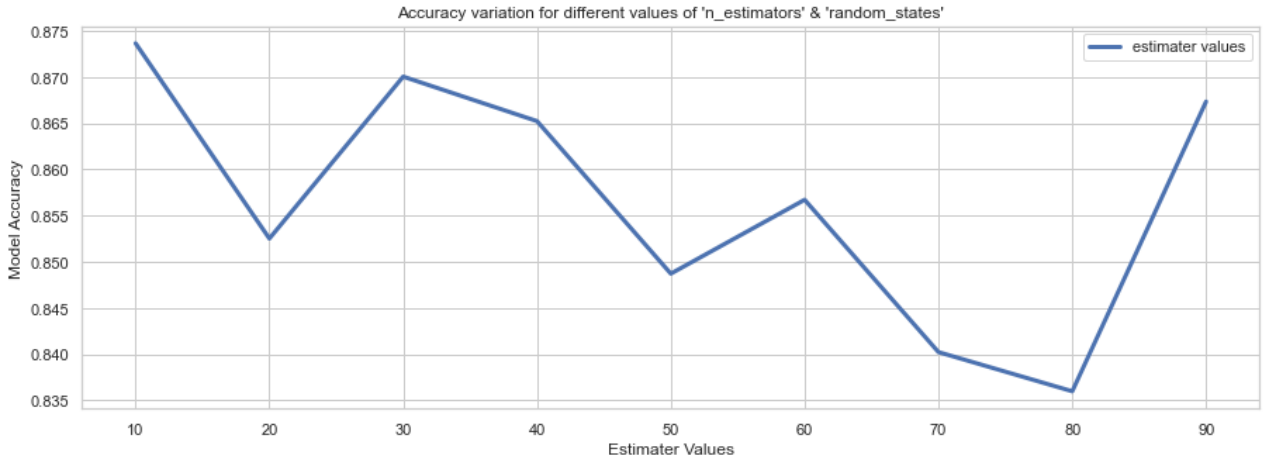
In [34]:
```
#Evaluate the RandomizedSerachCV model
rs_model.best_score_
```

Out[34]:
```
0.3510010746913461
```

Here it was given less than the previous accuracy value (0.840). So I had to try in random way to increase the accuracy

In [35]:
```
n_estimates_rndm_st = np.arange(10,100,10)
estimater_values = []

for i in n_estimates_rndm_st:
    rfr_selected_model = RandomForestRegressor(n_estimators=i, random_state=i, c
    rfr_pred = rfr_selected_model.predict(x_test)
    estimater_values.append(r2_score(y_test,rfr_pred))
```

In [37]:
```
plt.figure(figsize=(15,5))

sns.lineplot(x = n_estimates_rndm_st, y = estimater_values, label = 'estimater v
plt.title("Accuracy variation for different values of 'n_estimators' & 'random_s
plt.xlabel('Estimater Values')
plt.ylabel('Model Accuracy')
plt.show()
```



Conclusion

So finaly it can be fitted as follow for n_estimators value and random_satate 10.

In [38]:
```
rfr_selected_model = RandomForestRegressor(n_estimators=10, random_state=10, cri
rfr_pred = rfr_selected_model.predict(x_test)
print("Tuning model's Accuracy is " + str(round((r2_score(y_test,rfr_pred))* 100
```

```
Tuning model's Accuracy is 87.37%
```

In [ ]: