- Name : SAGNIK SAMANTA
- Internship ID : UMIP26929
- Internship : Data Analyst Intern at UnifiedMentor
- Project : Laptop Price Analysis

Introduction

Laptots are lightweight and Portable,making them easy to carry and use them anywhere,anytime.Laptops enable users to work,study and communicate from anywhere,leading to increased Productivity and efficiency.In today's rapidly evolving technological environment laptops are essential tools for individuals and businesses alike.Accurate prediction of laptop prices helps retailers develop effective pricing strategies and enables consumers to plan their budgets and select the most suitable laptops.

Project Description

We are provided with a Laptop dataset containing laptop specifications.In this Project we tried to build a machine learning model that can be used for predicting prices of laptops based on their features. This dataset contains numerical as well as categorical feature variables.Here we employed Multiple Linear Regression, Random Forest Regressor and XGboost to train the model and checked the accuracy of the model using metrics like Adjusted R-Squared,Root Mean Squared Error and Mean Absolute Error.Finally,we have plotted Actual Price and Predicted Price on the graph paper.

Column Description :

Provided Laptop dataset contains 1275 observations on 23 feature variables.The descriptions of the feature variables are stated below :

- Company : Laptop Manufacturer.
- Product : Brand and Model.
- TypeName : Laptop Type (Notebook, Ultrabook, Gaming, ...etc).
- Inches : Screen Size.
- Ram : Total amount of RAM in laptop (GBs).
- OS : Operating System installed.
- Weight : Laptop Weight in kilograms.
- Price_euros : Price of Laptop in Euros. (Target)
- Screen : screen definition (Standard, Full HD, 4K Ultra HD, Quad HD+).
- ScreenW : screen width (pixels).
- ScreenH : screen height (pixels).
- Touchscreen : whether or not the laptop has a touchscreen.
- IPSpanel: whether or not the laptop has an IPSpanel.
- RetinaDisplay : whether or not the laptop has retina display.
- CPU_company
- CPU_freq: frequency of laptop CPU (Hz).
- CPU_model
- PrimaryStorage : primary storage space (GB).

- PrimaryStorageType : primary storage type (HDD, SSD, Flash Storage, Hybrid).
- SecondaryStorage : secondary storage space if any (GB).
- SecondaryStorageType : secondary storage type (HDD, SSD, Hybrid, None).
- GPU_company
- GPU_model

In [1]:
```
## Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_erro
```

In [2]:
```
## Load the Dataset
df=pd.read_csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/laptop_pr
df.head()
```

Out[2]:

	Company	Product	TypeName	Inches	Ram	OS	Weight	Price_euros	Screen	Sc
0	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1339.69	Standard	
1	Apple	Macbook Air	Ultrabook	13.3	8	macOS	1.34	898.94	Standard	
2	HP	250 G6	Notebook	15.6	8	No OS	1.86	575.00	Full HD	
3	Apple	MacBook Pro	Ultrabook	15.4	16	macOS	1.83	2537.45	Standard	
4	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1803.60	Standard	

5 rows × 23 columns

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

Out[3]:
```
(1275, 23)
```

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

Out[4]:
```
Index(['Company', 'Product', 'TypeName', 'Inches', 'Ram', 'OS', 'Weight',
       'Price_euros', 'Screen', 'ScreenW', 'ScreenH', 'Touchscreen',
       'IPSpanel', 'RetinaDisplay', 'CPU_company', 'CPU_freq', 'CPU_mode
l',
       'PrimaryStorage', 'SecondaryStorage', 'PrimaryStorageType',
       'SecondaryStorageType', 'GPU_company', 'GPU_model'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1275 entries, 0 to 1274
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Company               1275 non-null   object 
 1   Product               1275 non-null   object 
 2   TypeName              1275 non-null   object 
 3   Inches                1275 non-null   float64
 4   Ram                   1275 non-null   int64  
 5   OS                    1275 non-null   object 
 6   Weight                1275 non-null   float64
 7   Price_euros           1275 non-null   float64
 8   Screen                1275 non-null   object 
 9   ScreenW               1275 non-null   int64  
 10  ScreenH               1275 non-null   int64  
 11  Touchscreen           1275 non-null   object 
 12  IPSpanel              1275 non-null   object 
 13  RetinaDisplay         1275 non-null   object 
 14  CPU_company           1275 non-null   object 
 15  CPU_freq              1275 non-null   float64
 16  CPU_model             1275 non-null   object 
 17  PrimaryStorage        1275 non-null   int64  
 18  SecondaryStorage      1275 non-null   int64  
 19  PrimaryStorageType    1275 non-null   object 
 20  SecondaryStorageType  1275 non-null   object 
 21  GPU_company           1275 non-null   object 
 22  GPU_model             1275 non-null   object 
dtypes: float64(4), int64(5), object(14)
memory usage: 229.2+ KB
```

In [6]:
```
## Checking for missing values
print(df.isnull().sum())
```

```
Company                 0
Product                 0
TypeName                0
Inches                  0
Ram                     0
OS                      0
Weight                  0
Price_euros             0
Screen                  0
ScreenW                 0
ScreenH                 0
Touchscreen             0
IPSpanel                0
RetinaDisplay           0
CPU_company             0
CPU_freq                0
CPU_model               0
PrimaryStorage          0
SecondaryStorage        0
PrimaryStorageType      0
SecondaryStorageType    0
GPU_company             0
GPU_model               0
dtype: int64
```

In [7]:
```
## Drop rows with missing values
df.dropna(inplace=True)
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
0
```

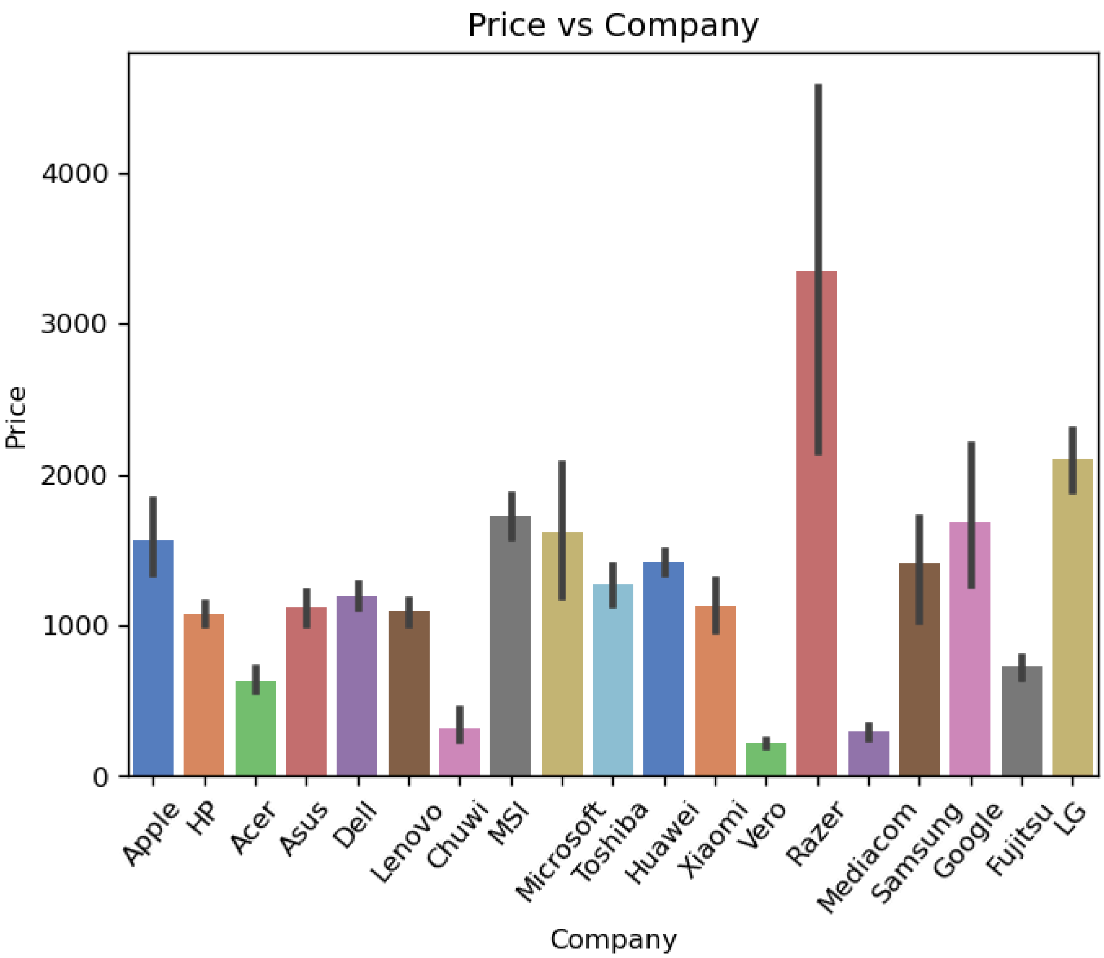
In [9]:
```
## Separating Categorical and Numerical values
catvars = df.select_dtypes(include=['object']).columns
numvars = df.select_dtypes(include = ['int32','int64','float32','float64']]

catvars,numvars
```

Out[9]:
```
(Index(['Company', 'Product', 'TypeName', 'OS', 'Screen', 'Touchscreen',
        'IPSpanel', 'RetinaDisplay', 'CPU_company', 'CPU_model',
        'PrimaryStorageType', 'SecondaryStorageType', 'GPU_company',
        'GPU_model'],
       dtype='object'),
 Index(['Inches', 'Ram', 'Weight', 'Price_euros', 'ScreenW', 'ScreenH',
        'CPU_freq', 'PrimaryStorage', 'SecondaryStorage'],
       dtype='object'))
```

In [10]:
```
df["Company"].value_counts()
```

Out[10]:
```
Company
Dell         291
Lenovo       289
HP           268
Asus         152
Acer         101
MSI           54
Toshiba       48
Apple         21
Samsung        9
Razer          7
Mediacom       7
Microsoft      6
Xiaomi         4
Vero           4
Chuwi          3
Google         3
Fujitsu        3
LG             3
Huawei         2
Name: count, dtype: int64
```

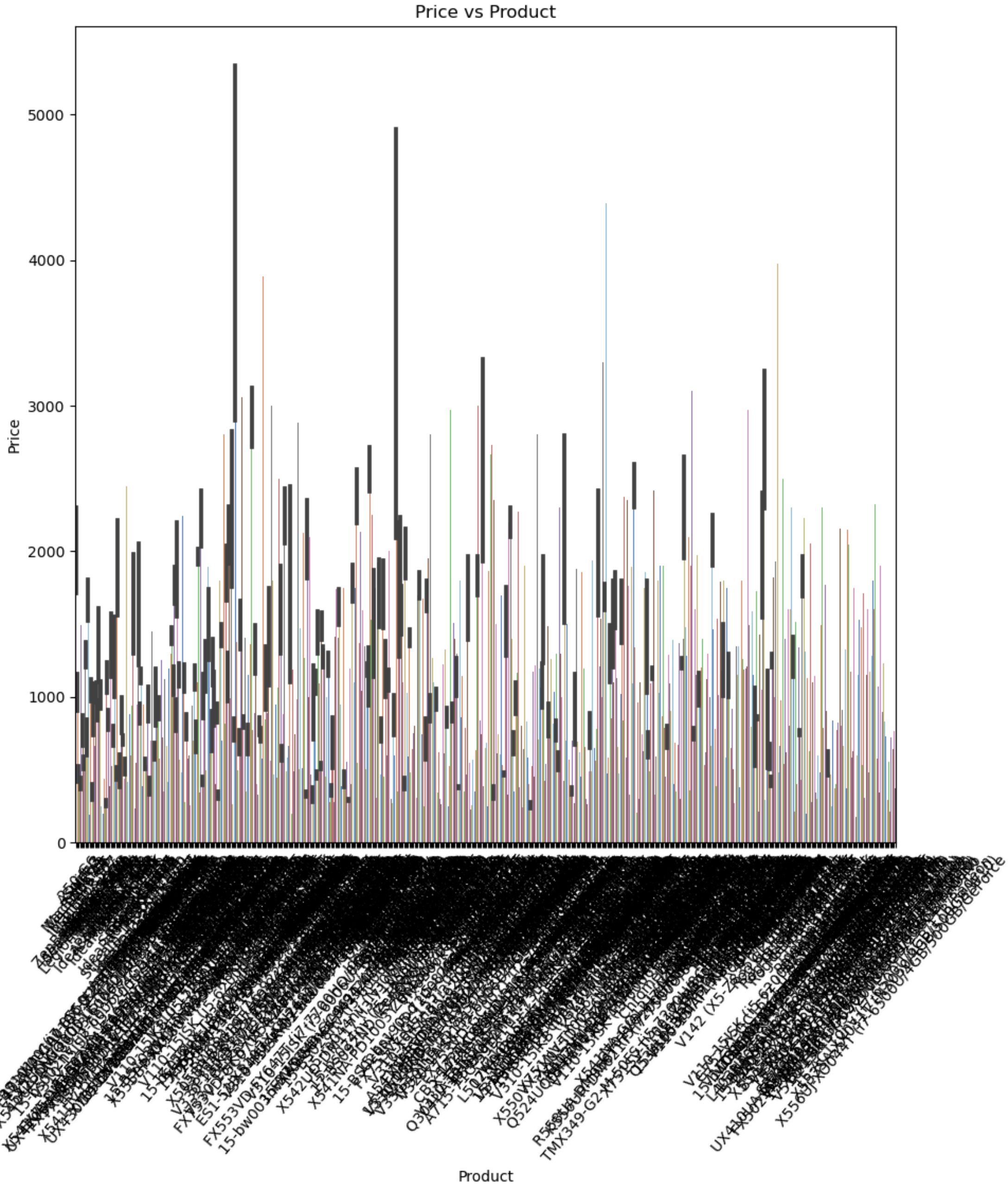
In [11]:
```
plt.figure(dpi=120)
sns.barplot(x="Company",y="Price_euros",palette="muted",data=df)
plt.xlabel("Company")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs Company")
plt.show()
```



In [12]:
```
df["Product"].value_counts()
```

Out[12]:
```
Product
XPS 13                                30
Inspiron 3567                         25
250 G6                                21
Vostro 3568                           19
Legion Y520-15IKBN                    19
                                      ..
VivoBook E201NA                        1
Ideapad 520-15IKBR                     1
Thinkpad X260                          1
Rog G752VL-UH71T                       1
X553SA-XX031T (N3050/4GB/500GB/W10)    1
Name: count, Length: 618, dtype: int64
```

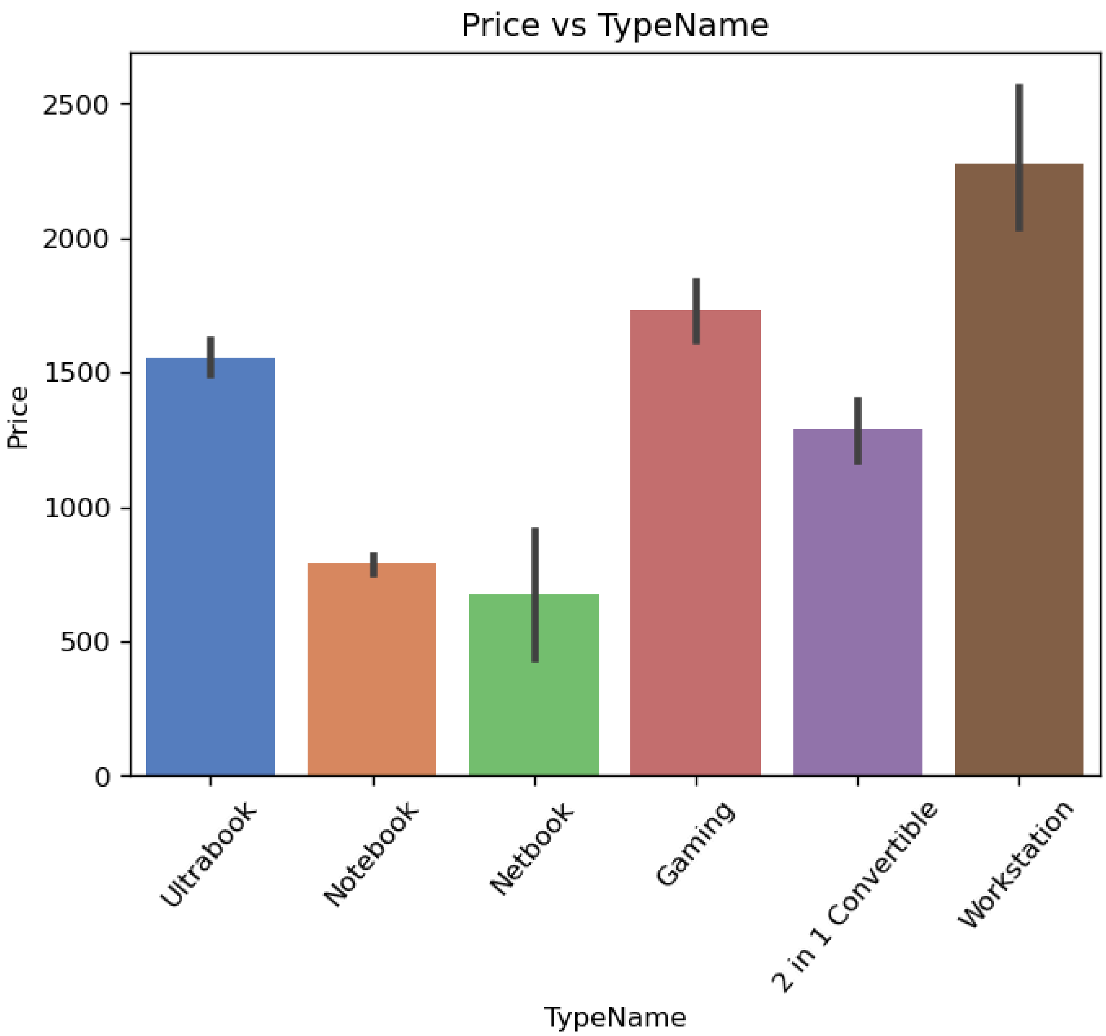
In [13]:
```
plt.figure(figsize=(10,10))
sns.barplot(x="Product",y="Price_euros",palette="muted",data=df)
plt.xlabel("Product")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs Product")
plt.show()
```



In [14]:
```
df["TypeName"].value_counts()
```

Out[14]:
```
TypeName
Notebook              707
Gaming                205
Ultrabook             194
2 in 1 Convertible    117
Workstation            29
Netbook                23
Name: count, dtype: int64
```

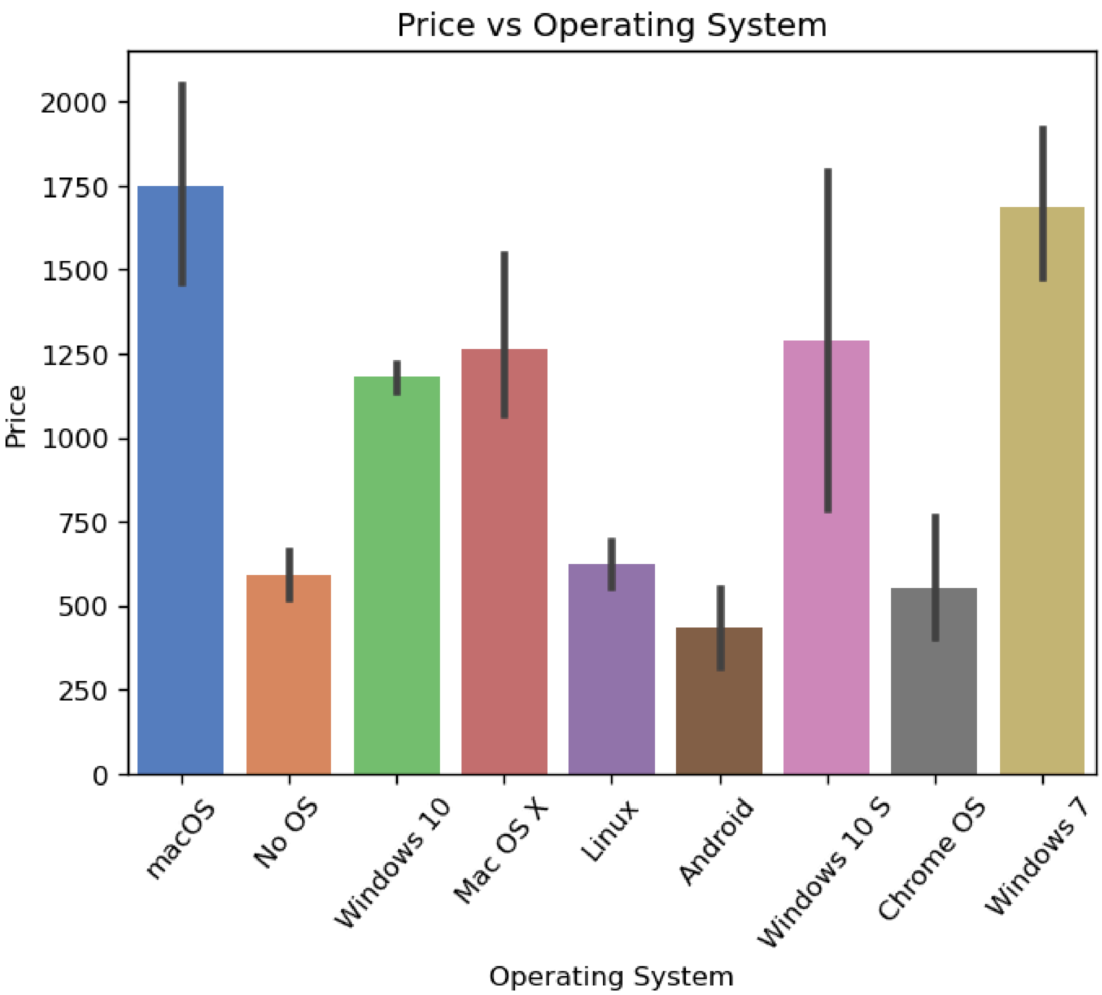
In [15]:
```
plt.figure(dpi=120)
sns.barplot(x="TypeName",y="Price_euros",palette="muted",data=df)
plt.xlabel("TypeName")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs TypeName")
plt.show()
```



In [16]:
```
df["OS"].value_counts()
```

Out[16]:
```
OS
Windows 10      1048
No OS             66
Linux             58
Windows 7         45
Chrome OS         27
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: count, dtype: int64
```

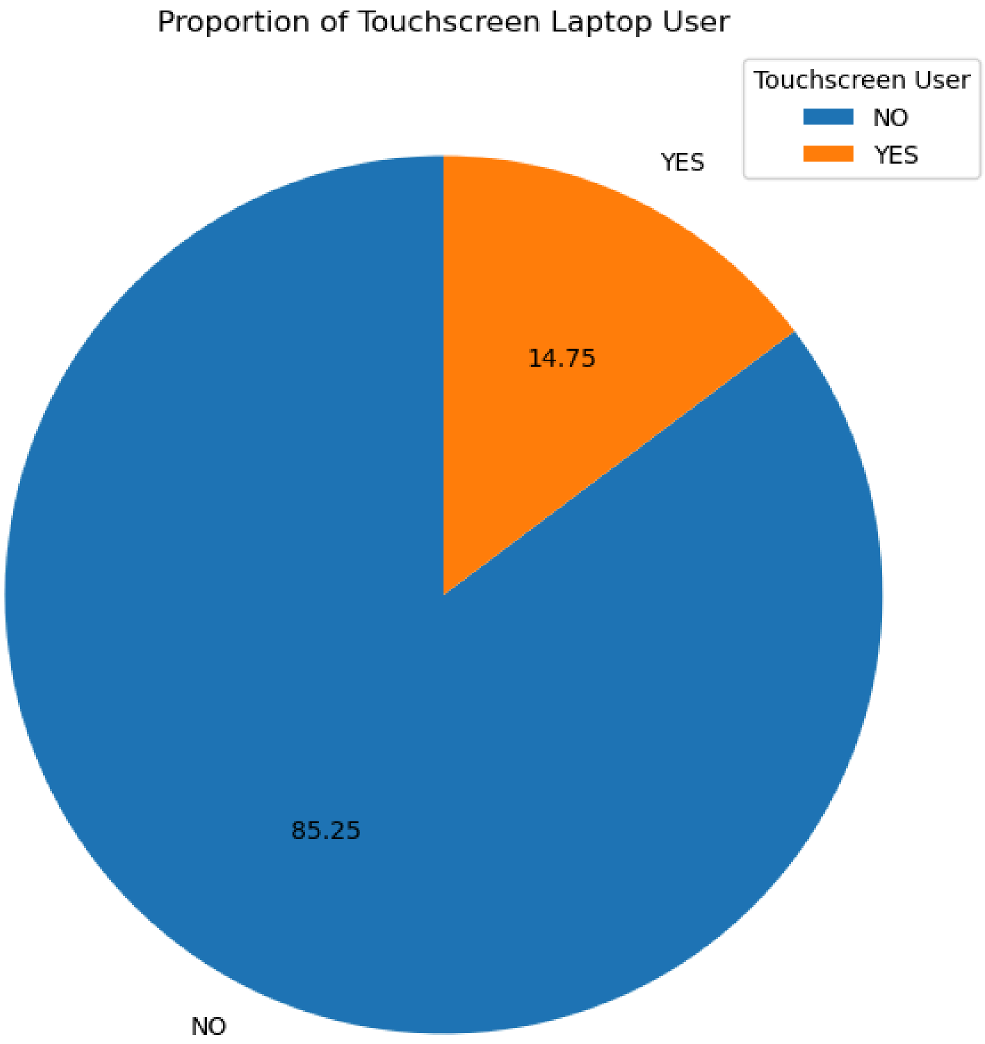
In [17]:
```
plt.figure(dpi=120)
sns.barplot(x="OS",y="Price_euros",palette="muted",data=df)
plt.xlabel("Operating System")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs Operating System")
plt.show()
```



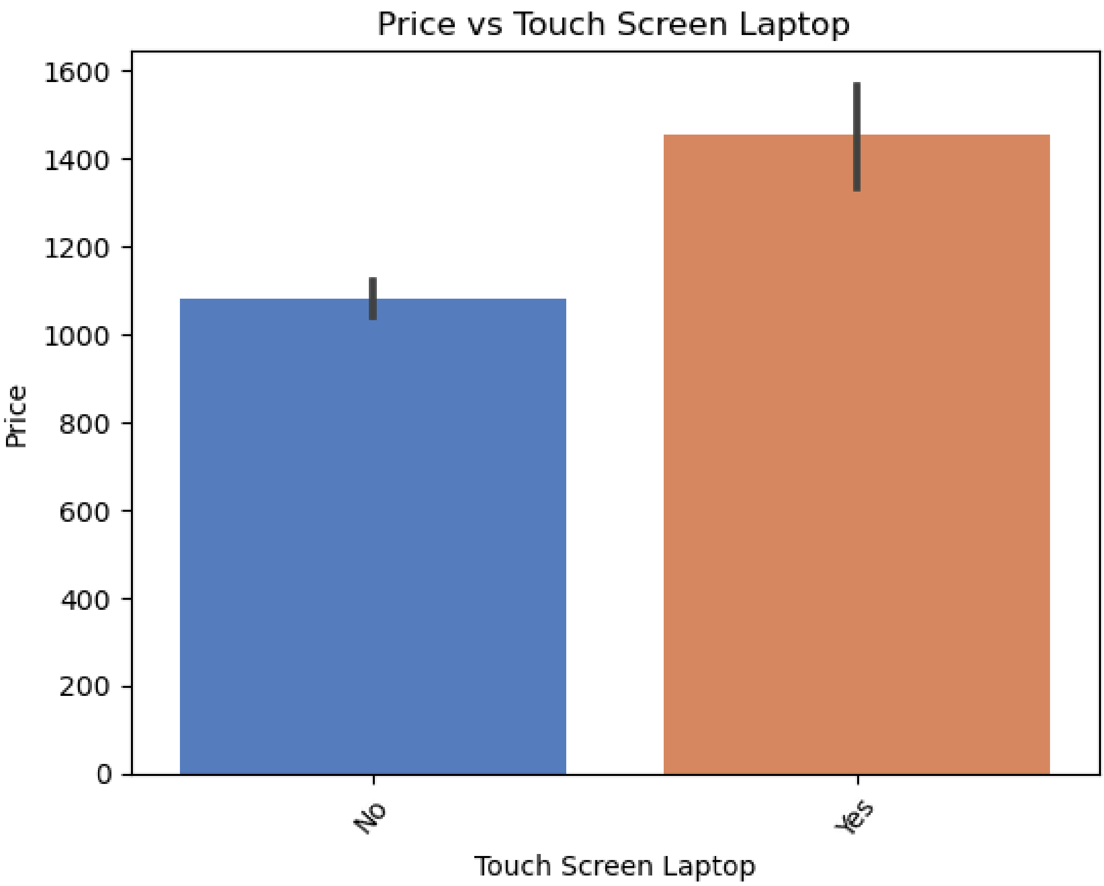
In [18]:
```
df["Touchscreen"].value_counts()
```

Out[18]:
```
Touchscreen
No     1087
Yes     188
Name: count, dtype: int64
```

In [19]:
```
plt.figure(figsize=(10,8))
plt.pie(df["Touchscreen"].value_counts(),labels=["NO","YES"],autopct="%0.2
plt.title("Proportion of Touchscreen Laptop User")
plt.legend(title="Touchscreen User",loc=1)
plt.show()
```



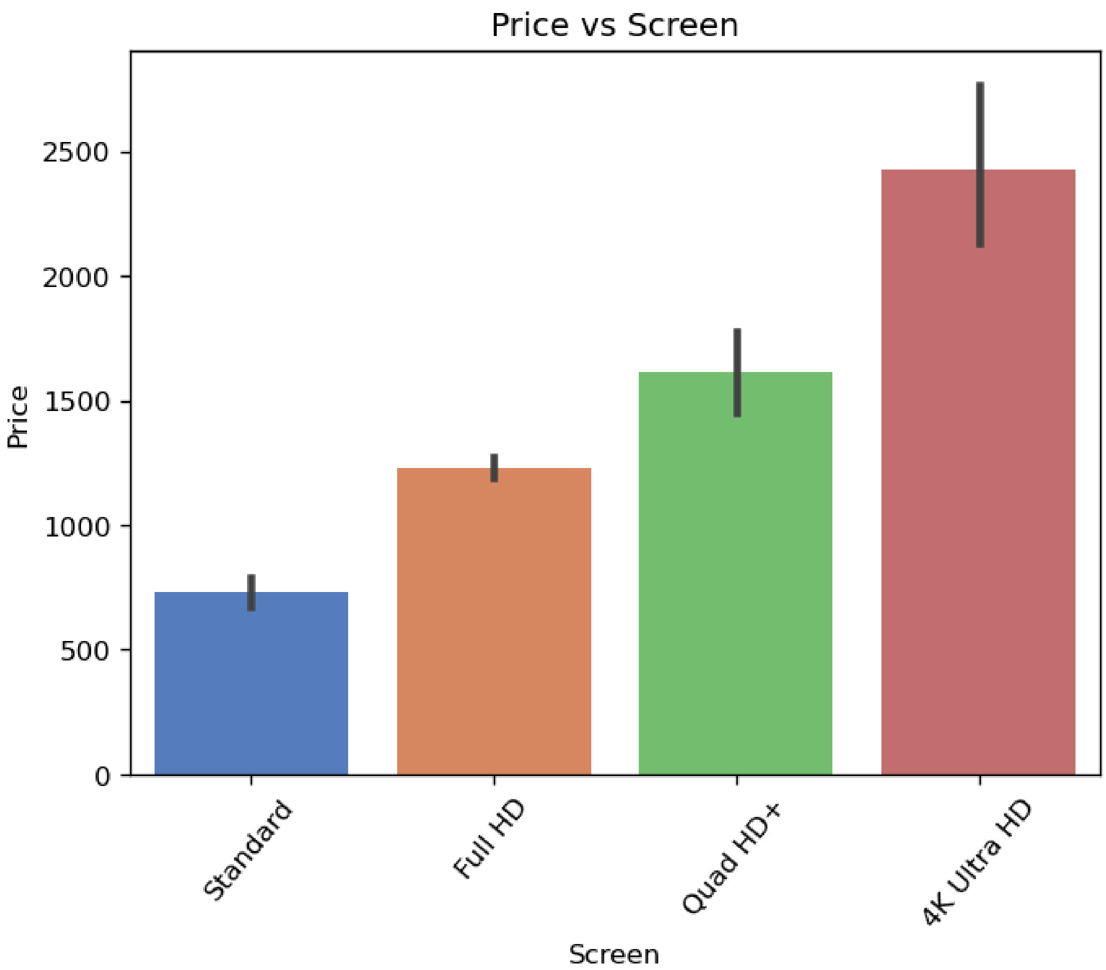
In [20]:
```
plt.figure(dpi=100)
sns.barplot(x="Touchscreen",y="Price_euros",palette="muted",data=df)
plt.xlabel("Touch Screen Laptop")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs Touch Screen Laptop")
plt.show()
```



In [21]:
```
df["Screen"].value_counts()
```

Out[21]:
```
Screen
Full HD         835
Standard        369
4K Ultra HD      43
Quad HD+         28
Name: count, dtype: int64
```

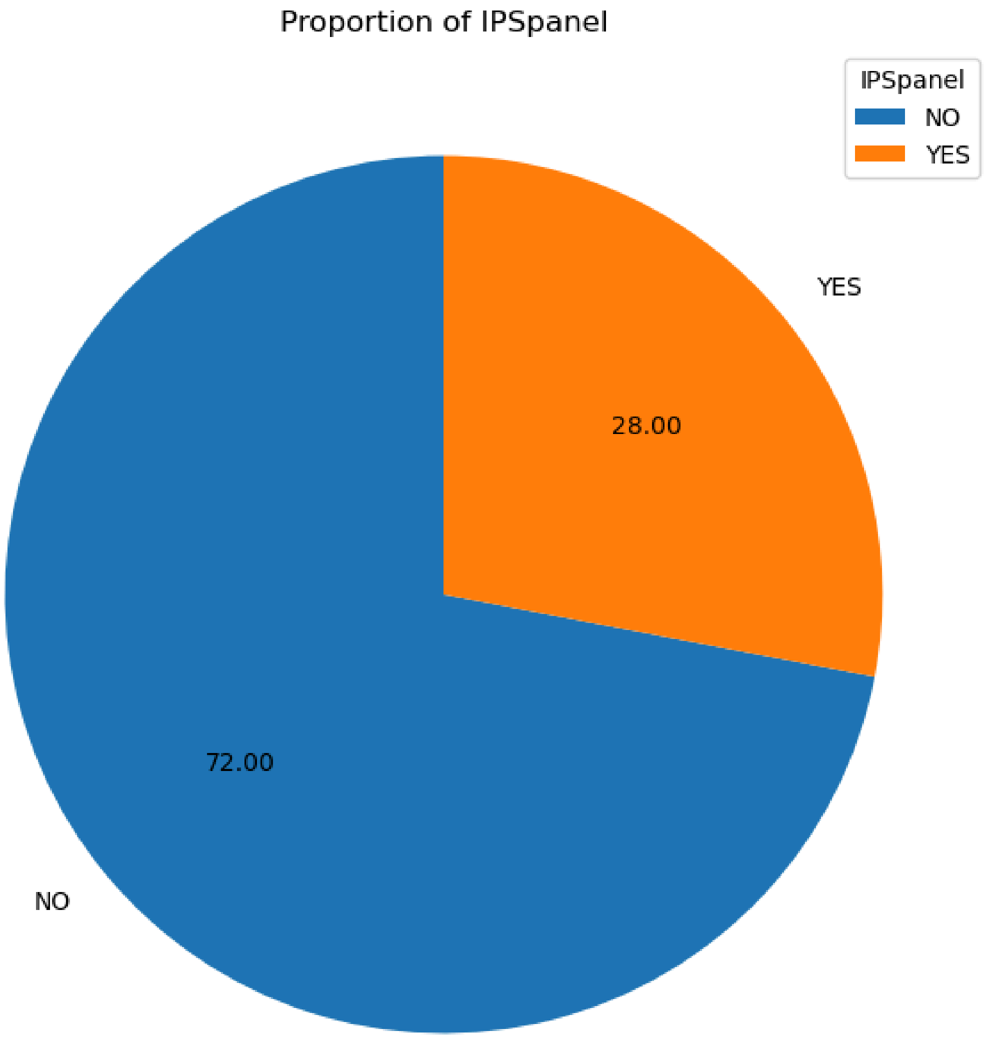
In [22]:
```
plt.figure(dpi=120)
sns.barplot(x="Screen",y="Price_euros",palette="muted",data=df)
plt.xlabel("Screen")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs Screen")
plt.show()
```



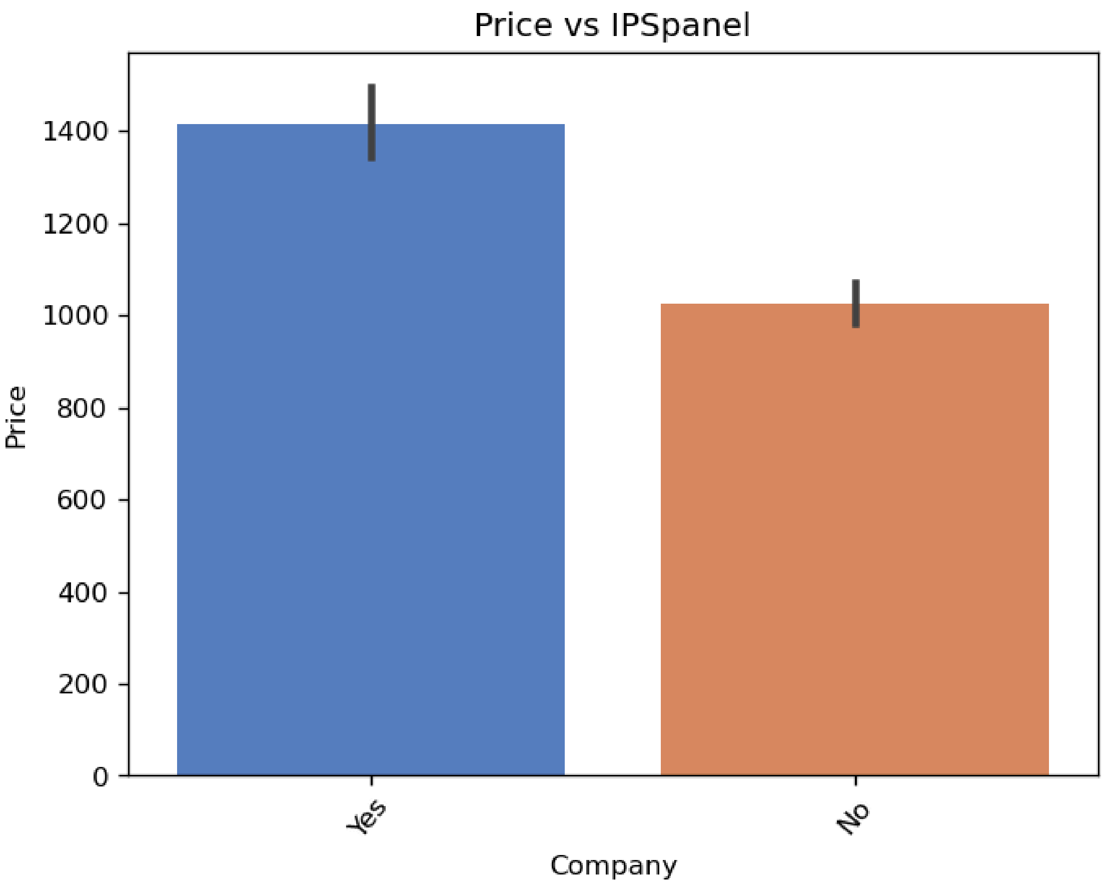
In [23]:
```
df["IPSpanel"].value_counts()
```

Out[23]:
```
IPSpanel
No     918
Yes    357
Name: count, dtype: int64
```

In [24]:
```
plt.figure(figsize=(10,8))
plt.pie(df["IPSpanel"].value_counts(),labels=["NO","YES"],autopct="%0.2f",s
plt.title("Proportion of IPSpanel")
plt.legend(title="IPSpanel",loc=1)
plt.show()
```



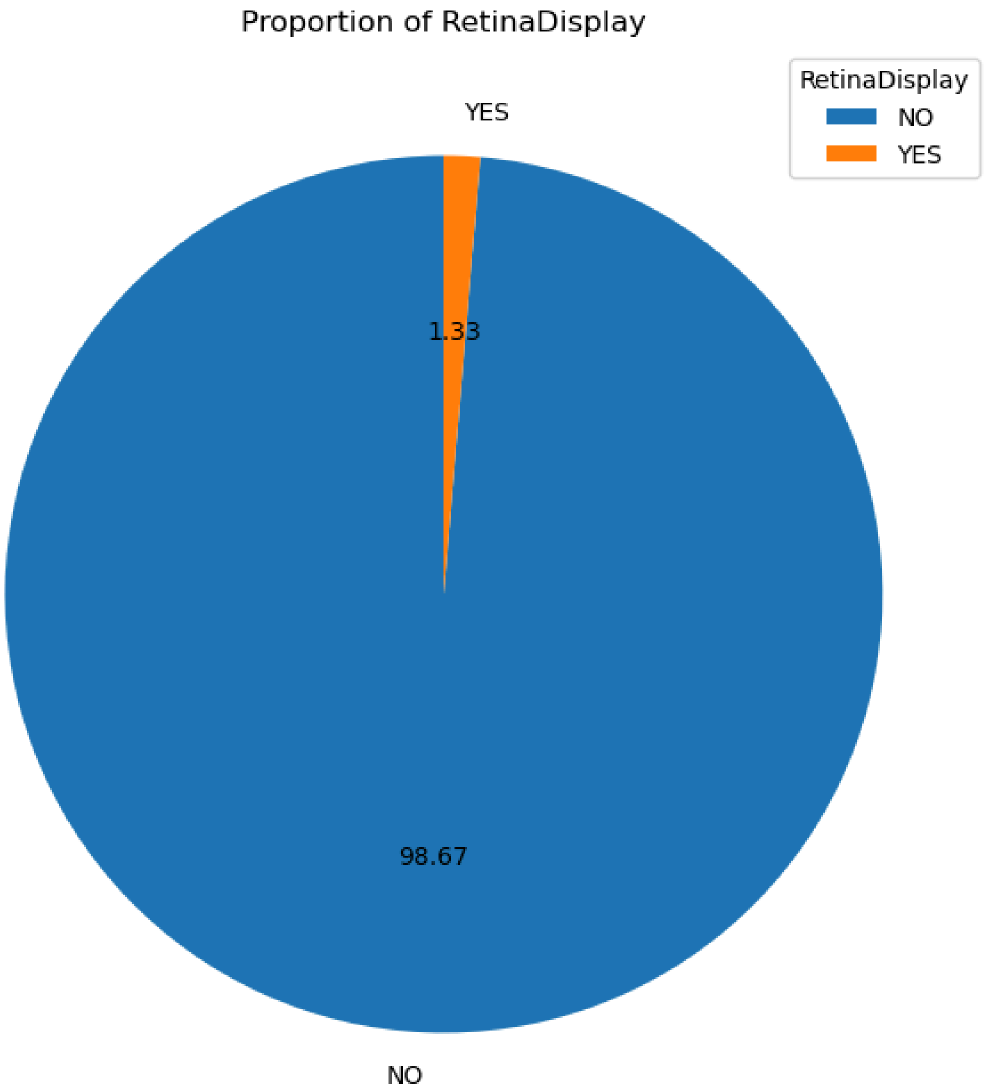
In [25]:
```
plt.figure(dpi=120)
sns.barplot(x="IPSpanel",y="Price_euros",palette="muted",data=df)
plt.xlabel("Company")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs IPSpanel")
plt.show()
```



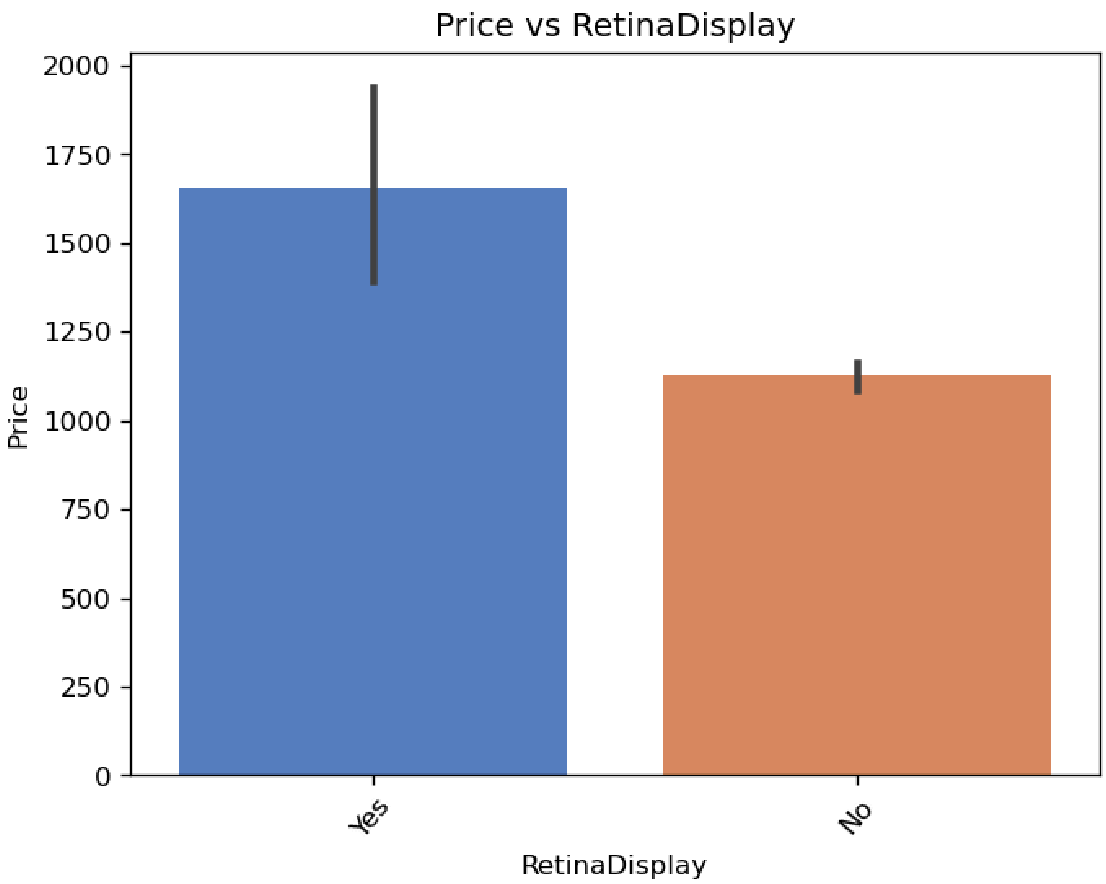
In [26]:
```
df["RetinaDisplay"].value_counts()
```

Out[26]:
```
RetinaDisplay
No     1258
Yes      17
Name: count, dtype: int64
```

In [27]:
```
plt.figure(figsize=(10,8))
plt.pie(df["RetinaDisplay"].value_counts(),labels=["NO","YES"],autopct="%0
plt.title("Proportion of RetinaDisplay")
plt.legend(title="RetinaDisplay",loc=1)
plt.show()
```



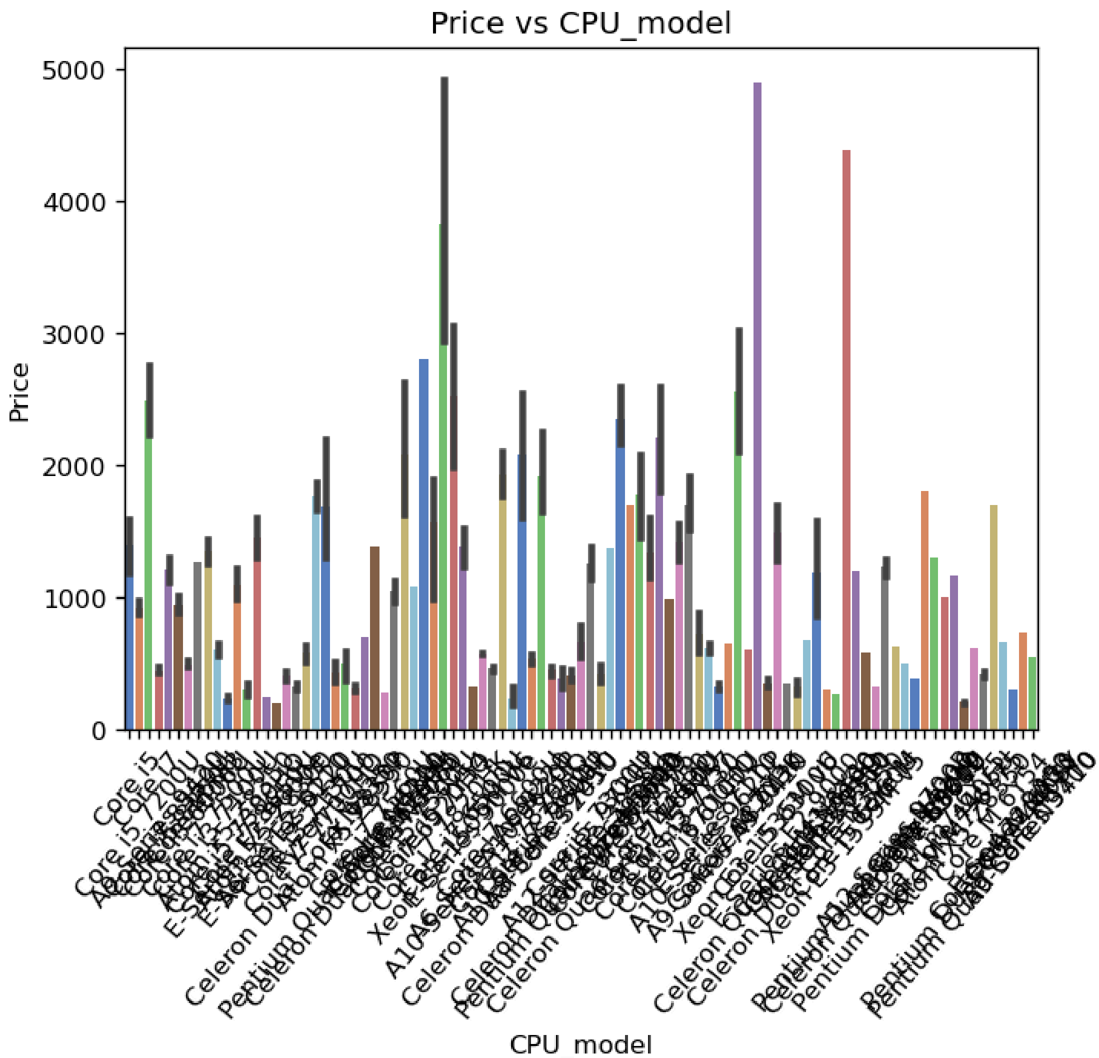
In [28]:
```
plt.figure(dpi=120)
sns.barplot(x="RetinaDisplay",y="Price_euros",palette="muted",data=df)
plt.xlabel("RetinaDisplay")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs RetinaDisplay")
plt.show()
```



In [29]:
```
df["CPU_model"].value_counts()
```

Out[29]:
```
CPU_model
Core i5 7200U       193
Core i7 7700HQ      147
Core i7 7500U       133
Core i3 6006U        81
Core i7 8550U        73
                   ...
Core M m3             1
E-Series E2-9000      1
Core M M3-6Y30        1
A6-Series 7310        1
A9-Series 9410        1
Name: count, Length: 93, dtype: int64
```

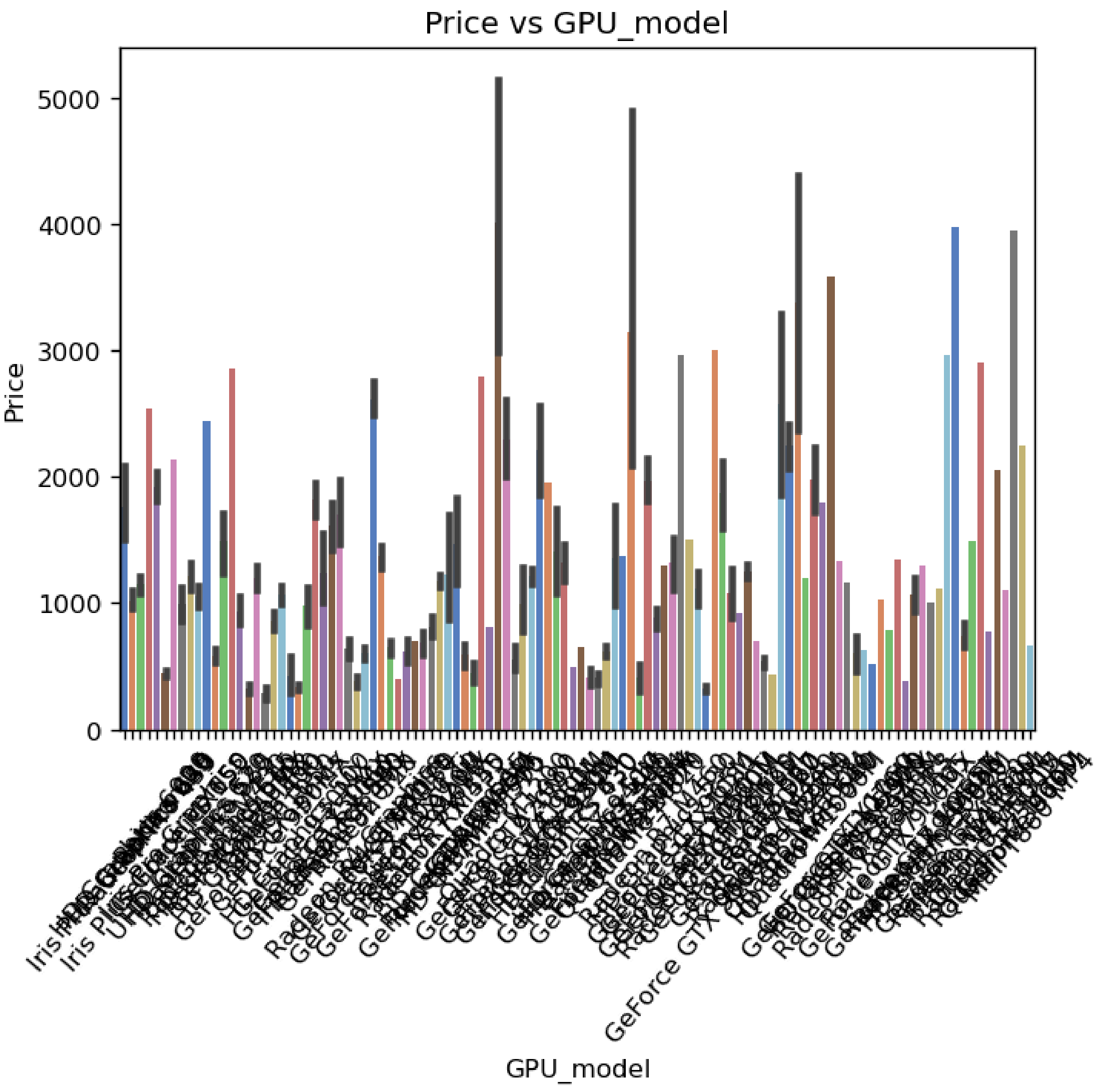
In [30]:
```
plt.figure(dpi=120)
sns.barplot(x="CPU_model",y="Price_euros",palette="muted",data=df)
plt.xlabel("CPU_model")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs CPU_model")
plt.show()
```



In [31]:
```
df["GPU_model"].value_counts()
```

Out[31]:
```
GPU_model
HD Graphics 620      279
HD Graphics 520      181
UHD Graphics 620      68
GeForce GTX 1050      66
GeForce GTX 1060      48
                    ...
Radeon R5 520          1
Radeon R7              1
HD Graphics 540        1
Radeon 540             1
Mali T860 MP4          1
Name: count, Length: 110, dtype: int64
```

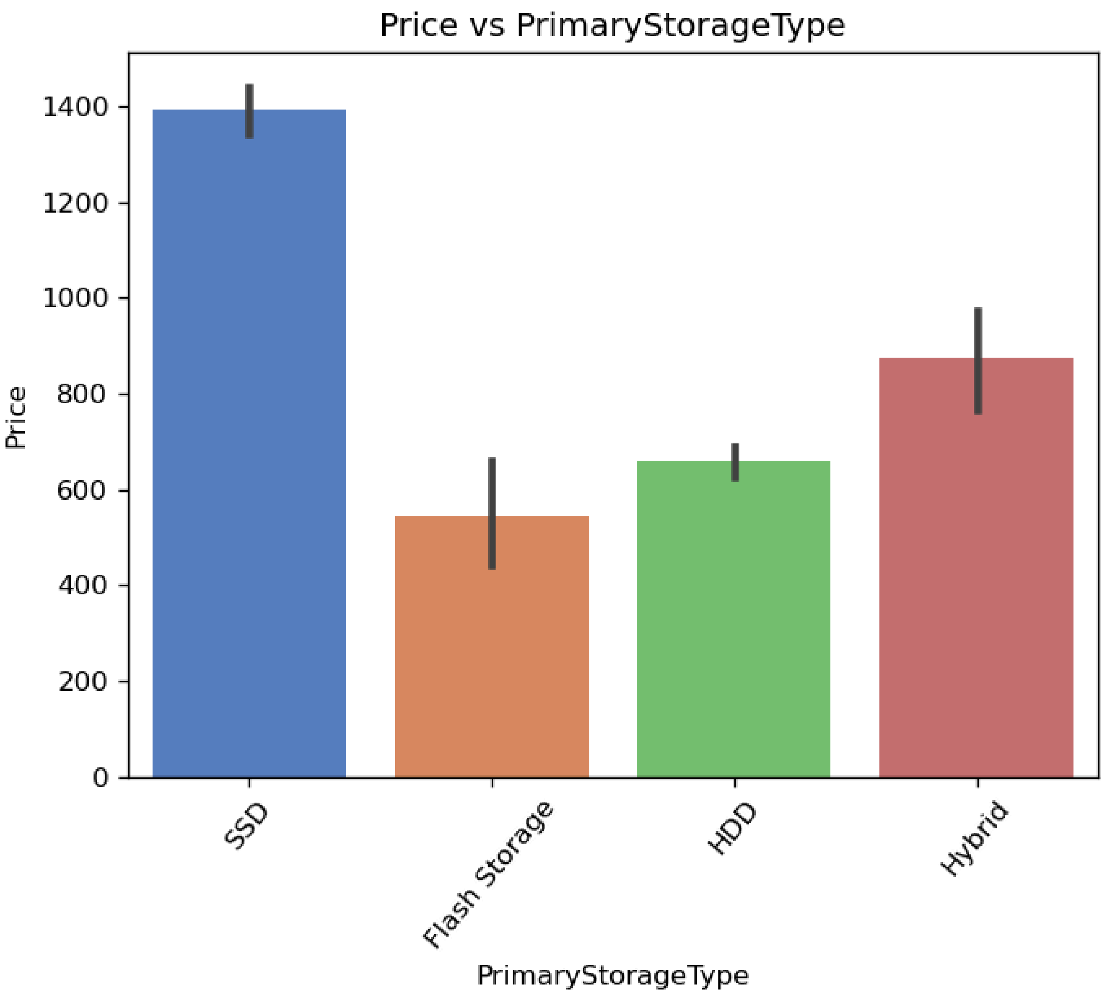
In [32]:
```
plt.figure(dpi=120)
sns.barplot(x="GPU_model",y="Price_euros",palette="muted",data=df)
plt.xlabel("GPU_model")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs GPU_model")
plt.show()
```



In [33]:
```
df["PrimaryStorageType"].value_counts()
```

Out[33]:
```
PrimaryStorageType
SSD              837
HDD              359
Flash Storage     71
Hybrid             8
Name: count, dtype: int64
```

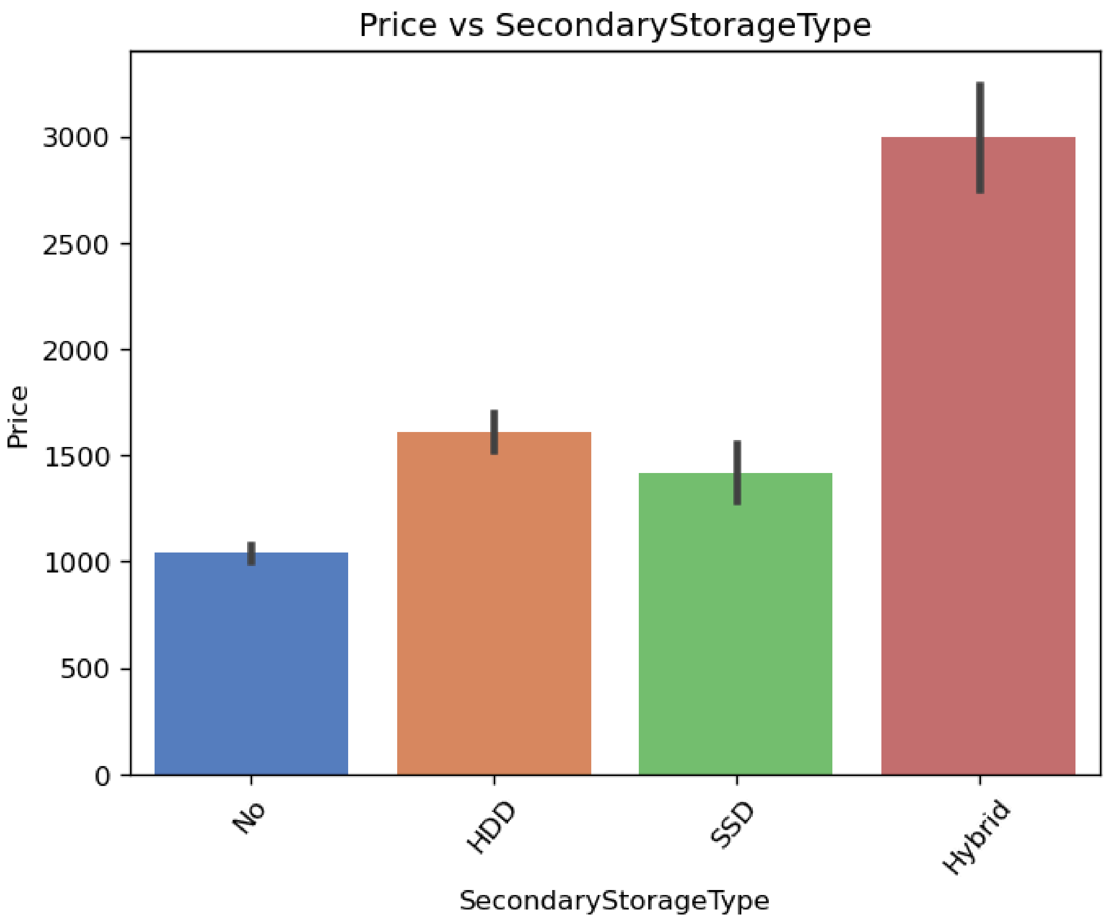
In [34]:
```
plt.figure(dpi=120)
sns.barplot(x="PrimaryStorageType",y="Price_euros",palette="muted",data=df
plt.xlabel("PrimaryStorageType")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs PrimaryStorageType")
plt.show()
```



In [35]:
```
df["SecondaryStorageType"].value_counts()
```

Out[35]:
```
SecondaryStorageType
No        1067
HDD        202
SSD          4
Hybrid       2
Name: count, dtype: int64
```

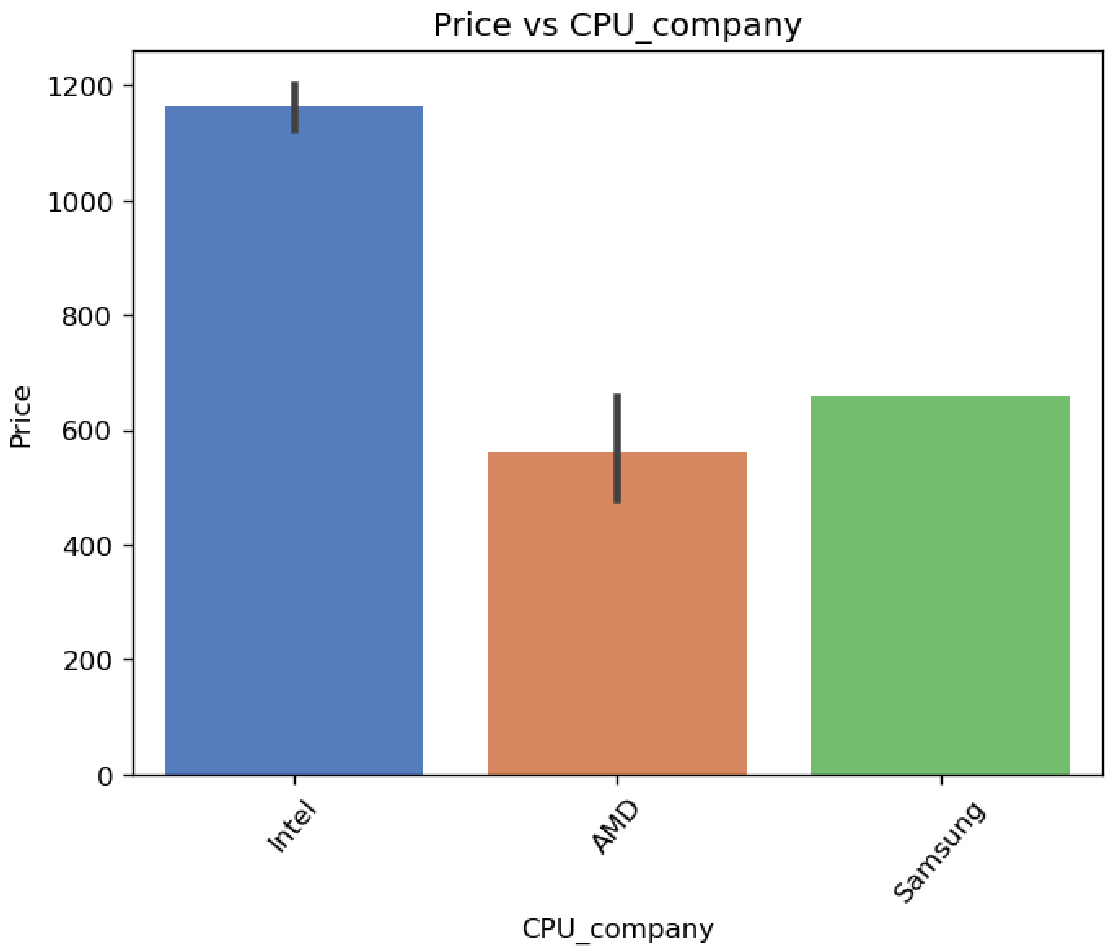
In [36]:
```
plt.figure(dpi=120)
sns.barplot(x="SecondaryStorageType",y="Price_euros",palette="muted",data=(
plt.xlabel("SecondaryStorageType")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs SecondaryStorageType")
plt.show()
```



In [37]:
```
df["CPU_company"].value_counts()
```

Out[37]:
```
CPU_company
Intel      1214
AMD          60
Samsung       1
Name: count, dtype: int64
```

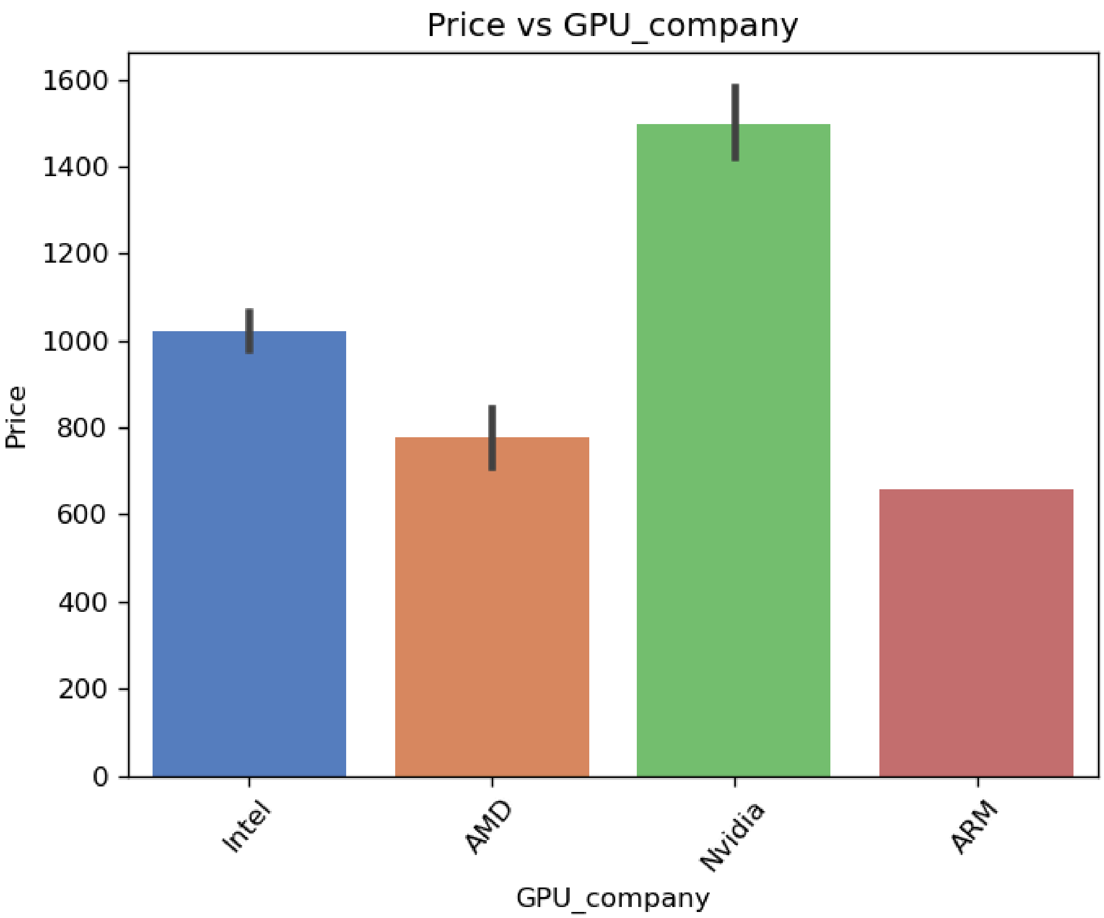
In [38]:
```
plt.figure(dpi=120)
sns.barplot(x="CPU_company",y="Price_euros",palette="muted",data=df)
plt.xlabel("CPU_company")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs CPU_company")
plt.show()
```



In [39]:
```
df["GPU_company"].value_counts()
```

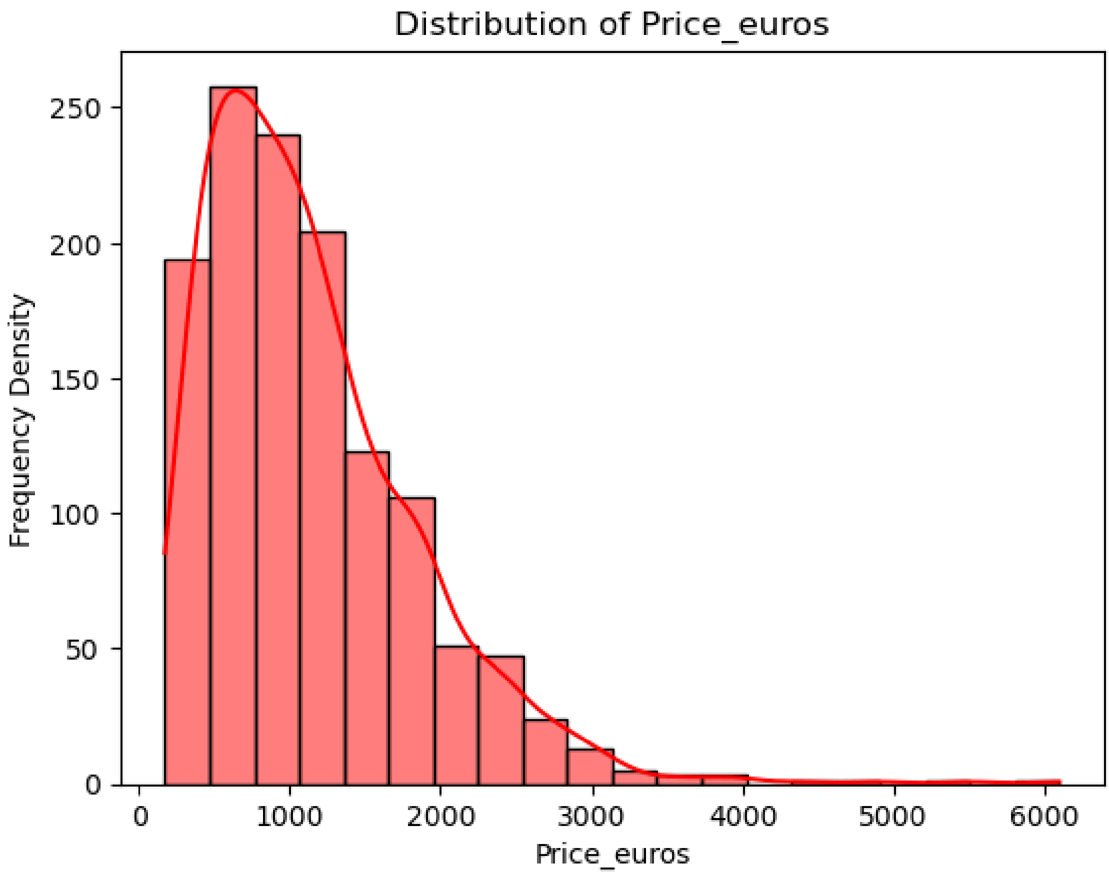
Out[39]:
```
GPU_company
Intel      704
Nvidia     396
AMD        174
ARM          1
Name: count, dtype: int64
```

In [40]:
```
plt.figure(dpi=120)
sns.barplot(x="GPU_company",y="Price_euros",palette="muted",data=df)
plt.xlabel("GPU_company")
plt.ylabel("Price")
plt.xticks(rotation=50)
plt.title("Price vs GPU_company")
plt.show()
```

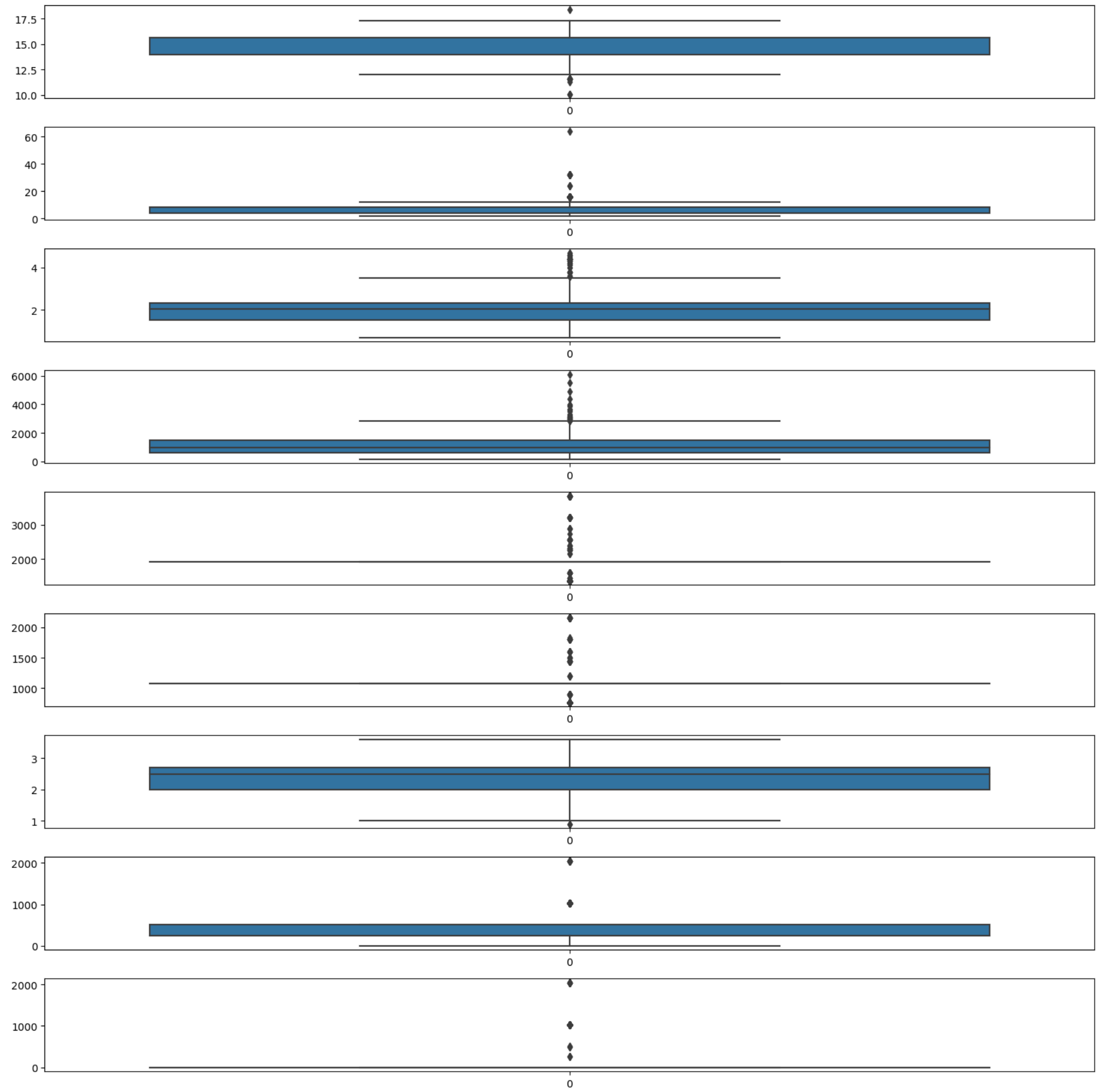


Visualizing Parameters

In [41]:
```
plt.figure(dpi=100)
sns.histplot(x="Price_euros",kde=True,color="r",bins=20,data=df)
plt.xlabel("Price_euros")
plt.ylabel("Frequency Density")
plt.title("Distribution of Price_euros")
plt.show()
```



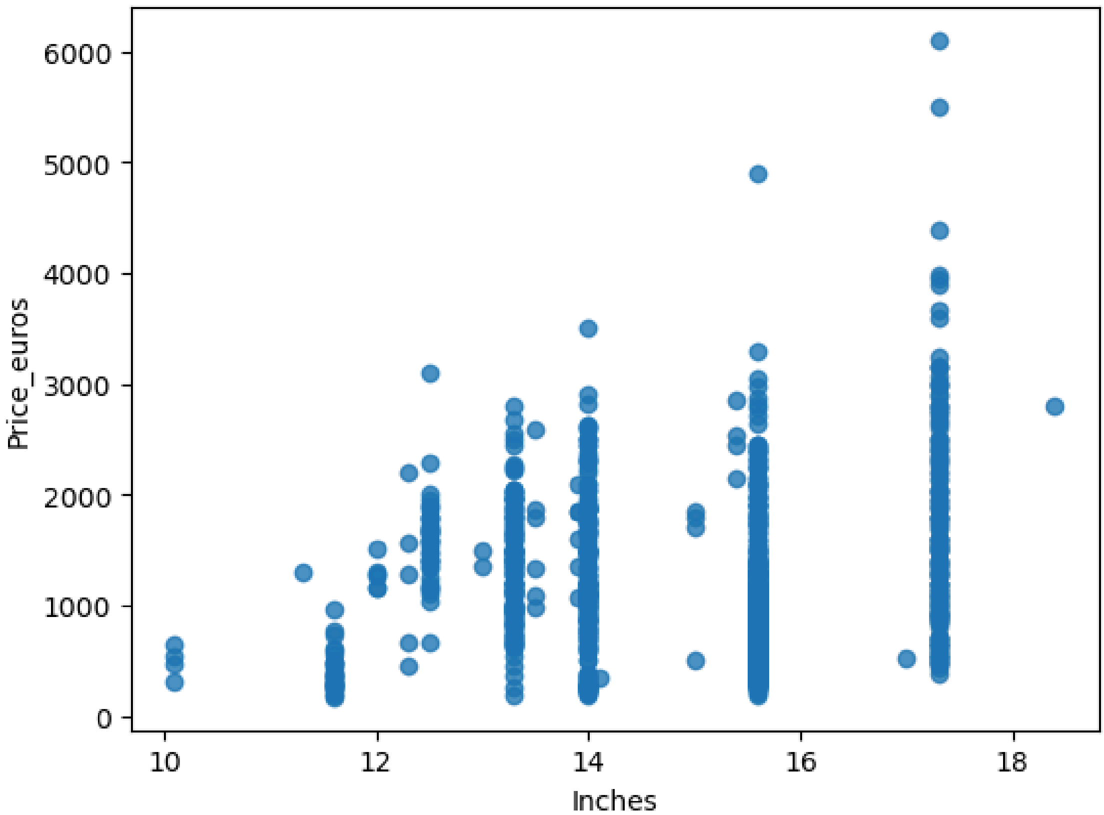
In [42]:
```
# Outlier Analysis
fig, axs = plt.subplots(9, figsize = (15,15))
plt1 = sns.boxplot(df['Inches'], ax = axs[0])
plt2 = sns.boxplot(df['Ram'], ax = axs[1])
plt3 = sns.boxplot(df['Weight'], ax = axs[2])
plt4 = sns.boxplot(df['Price_euros'], ax = axs[3])
plt5 = sns.boxplot(df['ScreenW'], ax = axs[4])
plt6 = sns.boxplot(df['ScreenH'], ax = axs[5])
plt7 = sns.boxplot(df['CPU_freq'], ax = axs[6])
plt8 = sns.boxplot(df['PrimaryStorage'], ax = axs[7])
plt9 = sns.boxplot(df['SecondaryStorage'], ax = axs[8])
plt.tight_layout()
```



From the above Price distribution graph we can see that Laptop with the lower price has a high demand compared to branded one.

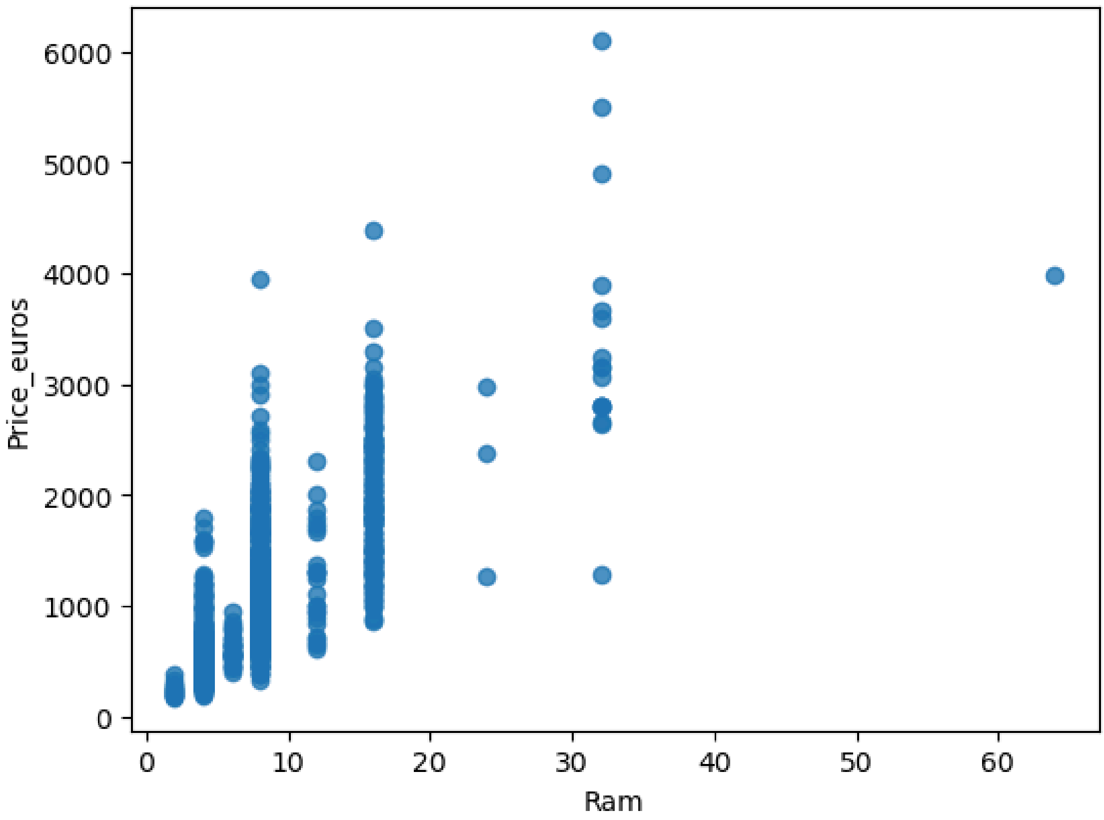
In [43]:
```
## Inches vs Price_euros
sns.regplot(x="Inches",y="Price_euros",scatter=True,fit_reg=False,data=df)
```

Out[43]: <Axes: xlabel='Inches', ylabel='Price_euros'>



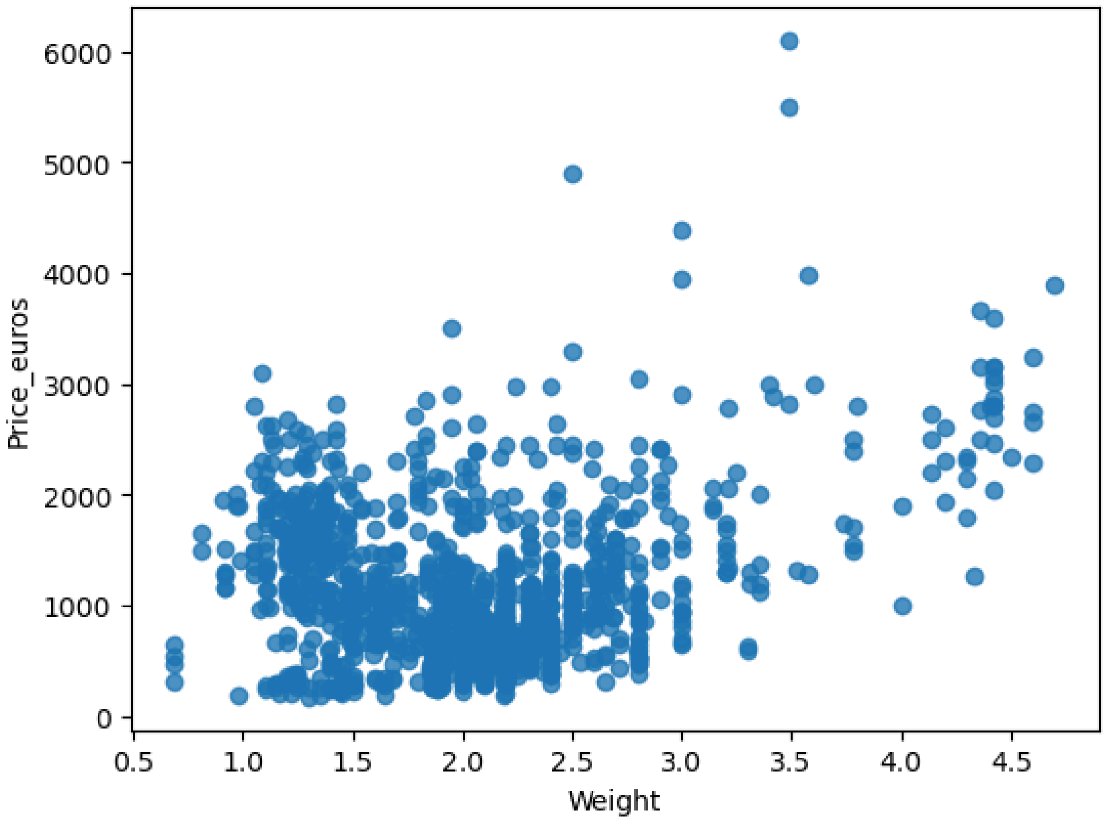
In [44]:
```
## Ram vs Price_euros
sns.regplot(x="Ram",y="Price_euros",scatter=True,fit_reg=False,data=df)
```

Out[44]: <Axes: xlabel='Ram', ylabel='Price_euros'>



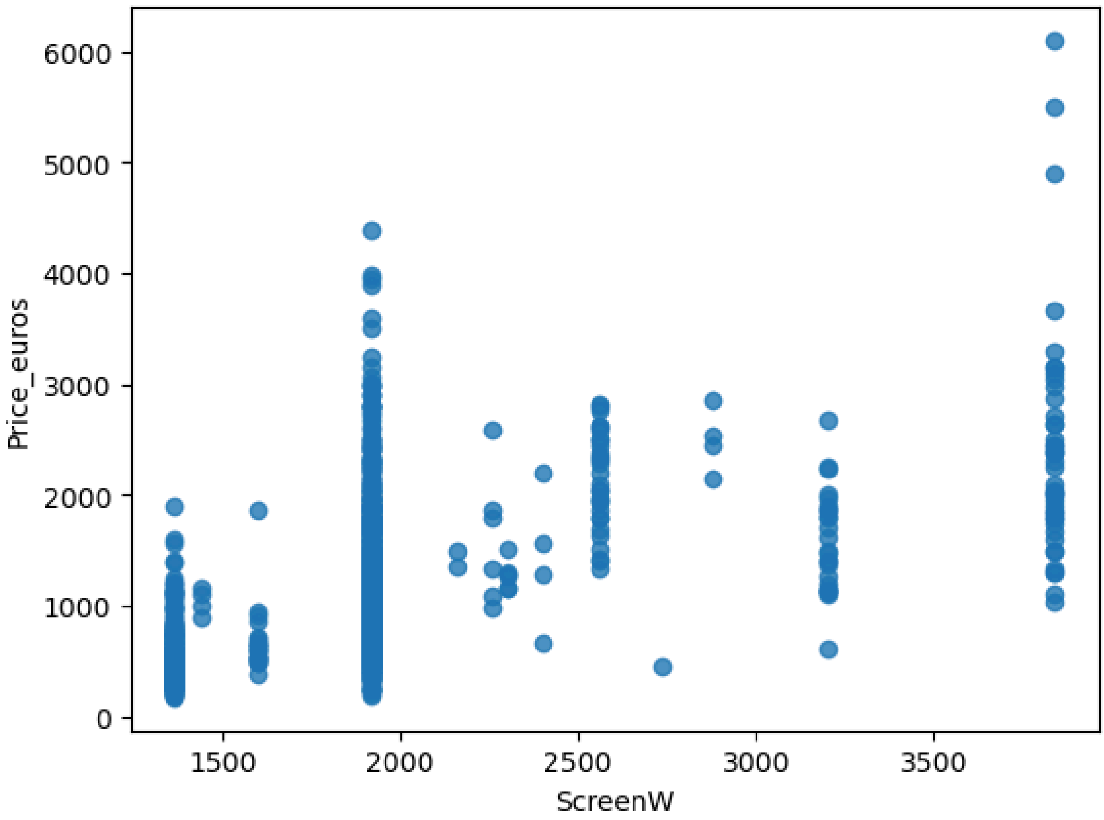
In [45]:
```
## Weight vs Price_euros
sns.regplot(x="Weight",y="Price_euros",scatter=True,fit_reg=False,data=df)
```

Out[45]: <Axes: xlabel='Weight', ylabel='Price_euros'>



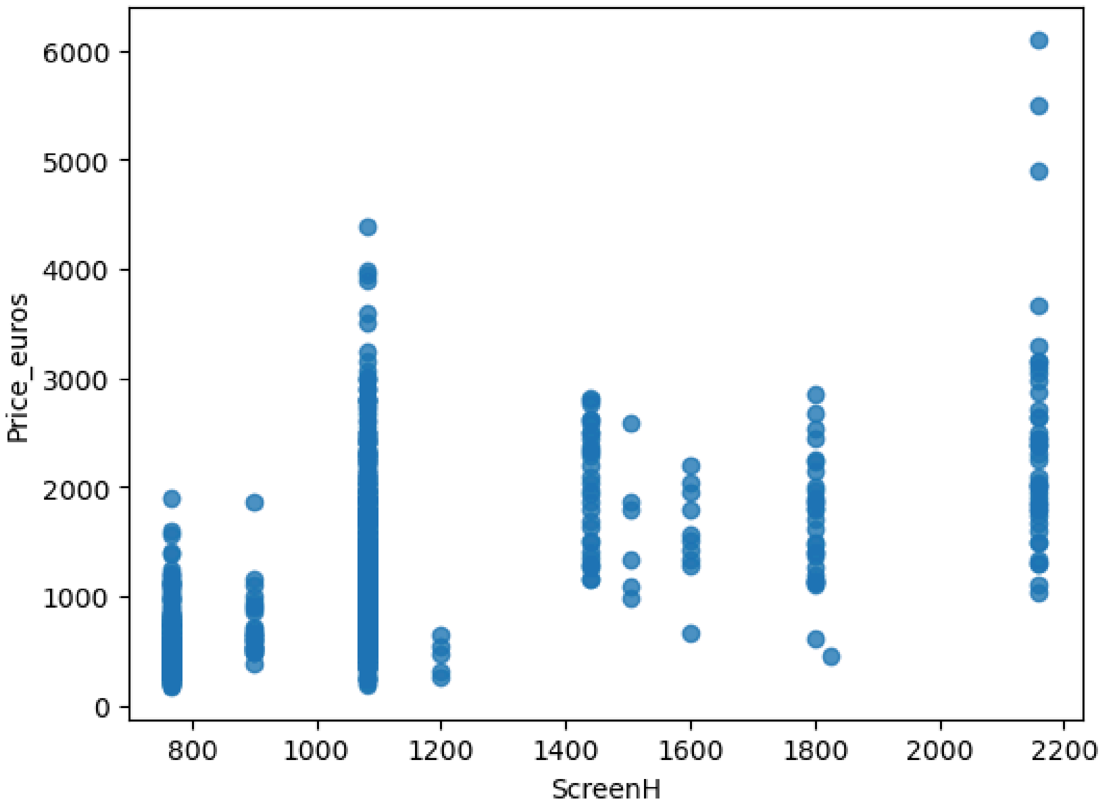
In [46]:
```
## ScreenW vs Price_euros
sns.regplot(x="ScreenW",y="Price_euros",scatter=True,fit_reg=False,data=df
```

Out[46]: <Axes: xlabel='ScreenW', ylabel='Price_euros'>



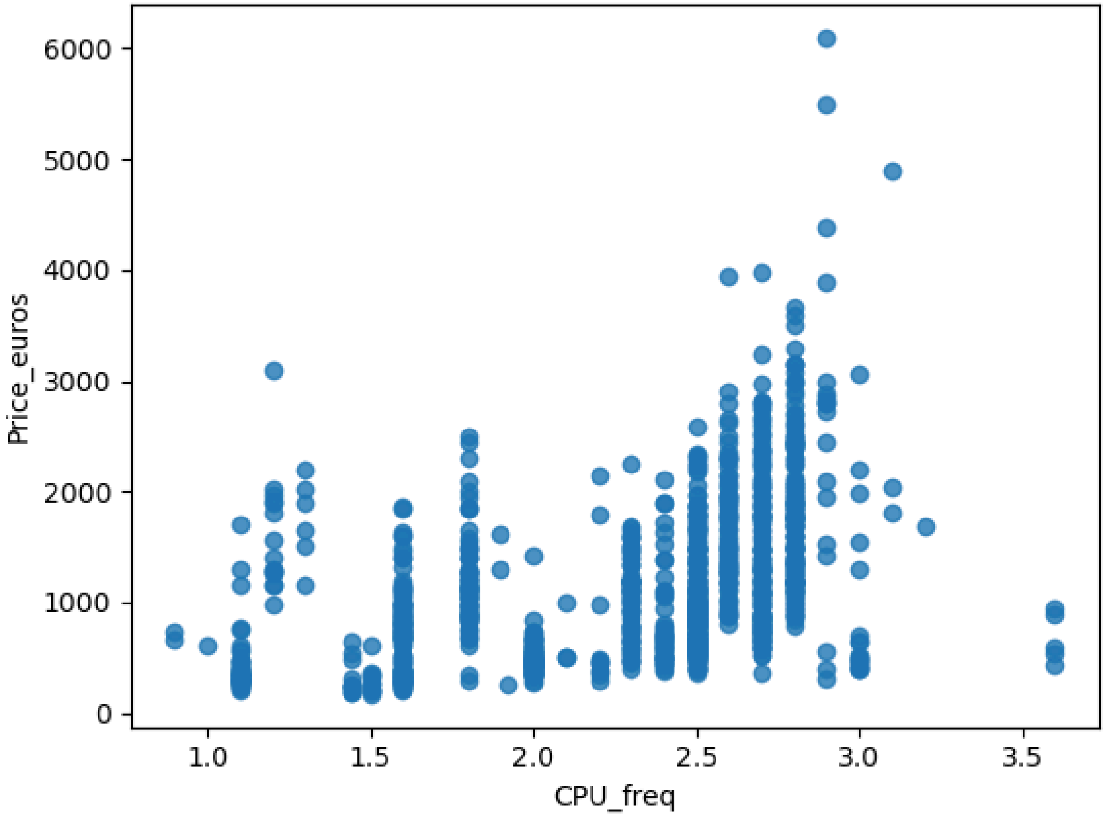
In [47]:
```
## ScreenH vs Price_euros
sns.regplot(x="ScreenH",y="Price_euros",scatter=True,fit_reg=False,data=df
```

Out[47]: <Axes: xlabel='ScreenH', ylabel='Price_euros'>



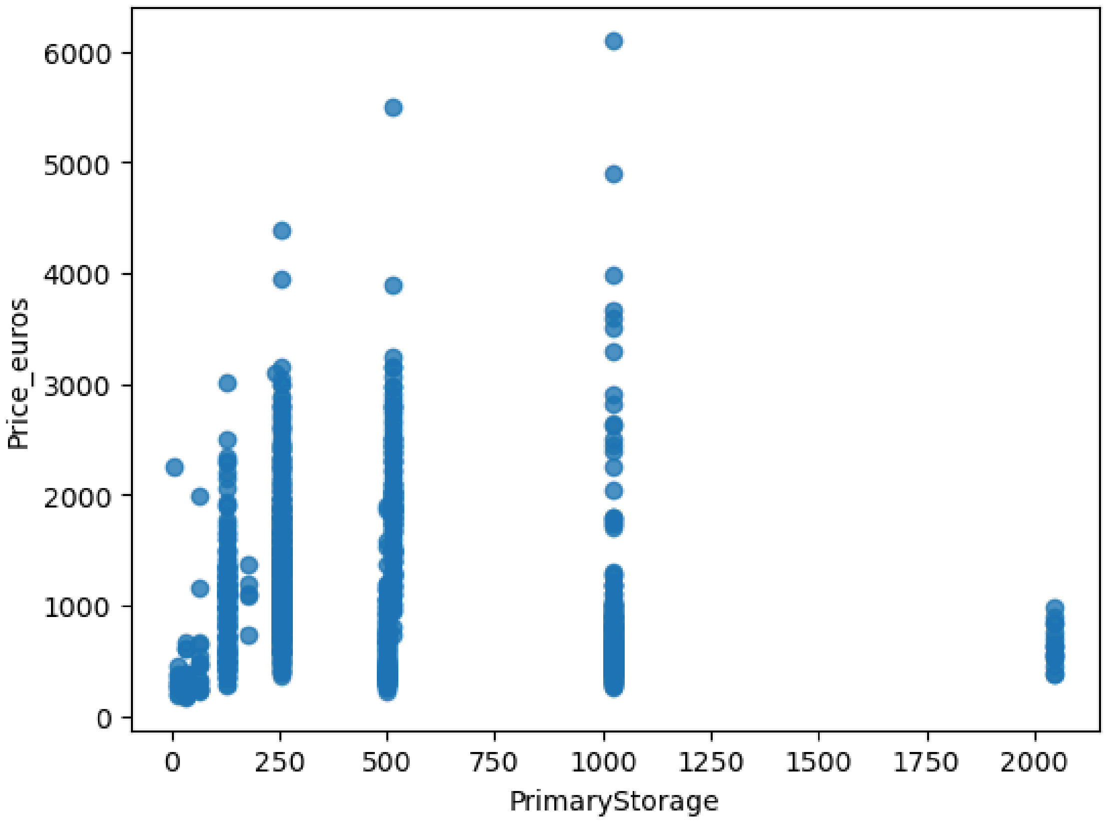
In [48]:
```
## CPU_freq vs Price_euros
sns.regplot(x="CPU_freq",y="Price_euros",scatter=True,fit_reg=False,data=d
```

Out[48]: <Axes: xlabel='CPU_freq', ylabel='Price_euros'>



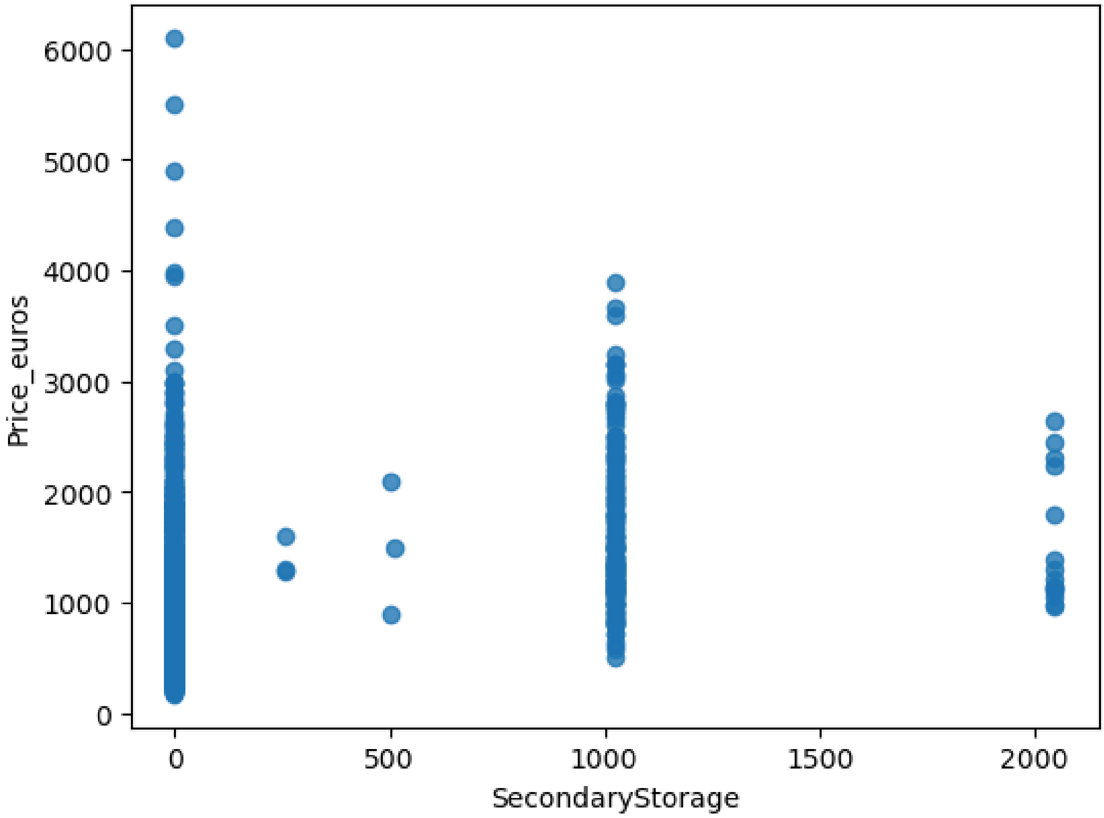
In [49]:
```
## PrimaryStorage vs Price_euros
sns.regplot(x="PrimaryStorage",y="Price_euros",scatter=True,fit_reg=False,(
```

Out[49]:
```
<Axes: xlabel='PrimaryStorage', ylabel='Price_euros'>
```



In [50]:
```
## SecondaryStorage vs Price_euros
sns.regplot(x="SecondaryStorage",y="Price_euros",scatter=True,fit_reg=Fals
```

Out[50]: <Axes: xlabel='SecondaryStorage', ylabel='Price_euros'>



Before proceeding to analysis we need to convert categorical variables into numerical variables using Label Encoding or One-Hot Encoding.

In [51]:
```
## Convert categorical variables to numerical variables
le = LabelEncoder()
df['Company'] = le.fit_transform(df['Company'])
df['Product'] = le.fit_transform(df['Product'])
df['TypeName'] = le.fit_transform(df['TypeName'])
df['OS'] = le.fit_transform(df['OS'])
df['Screen'] = le.fit_transform(df['Screen'])
df['RetinaDisplay'] = le.fit_transform(df['RetinaDisplay'])
df['CPU_company'] = le.fit_transform(df['CPU_company'])
df['GPU_company'] = le.fit_transform(df['GPU_company'])
df['PrimaryStorageType'] = le.fit_transform(df['PrimaryStorageType'])
df['SecondaryStorageType'] = le.fit_transform(df['SecondaryStorageType'])
df['GPU_model'] = le.fit_transform(df['GPU_model'])
df['CPU_model'] = le.fit_transform(df['CPU_model'])
df['Touchscreen'] = le.fit_transform(df['Touchscreen'])
df['IPSpanel'] = le.fit_transform(df['IPSpanel'])
```

Train Test split

```
We need to select our Feature Variables(Independent Variables) and
Target variable(Response Variable).Thereafter split the dataset into
training and testing dataset to evaluate the model's performance.
```

In [52]:
```
X=df.drop("Price_euros",axis=1)
y=df["Price_euros"]
```

In [53]:
```
## Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [54]:
```
## Scale the data using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Multiple Linear Regression Model

In [55]:
```
## Create a multiple linear regression model
ln_reg = LinearRegression()

## Train the model
ln_reg.fit(X_train_scaled, y_train)
```

Out[55]:
```
▾ LinearRegression
LinearRegression()
```

In [56]:
```
## Make predictions on test dataset
y_pred = ln_reg.predict(X_test_scaled)
```

In [57]:
```
## Evaluate the Model Performance
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error :{mae:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Absolute Error :257.81
Root Mean Squared Error: 358.02
R-squared: 0.74
```

In [58]:
```
## Actual Price and the Predicted Price
reg_model_diff = pd.DataFrame({'Actual Laptop Price': y_test, 'Predicted La
reg_model_diff
```

Out[58]:

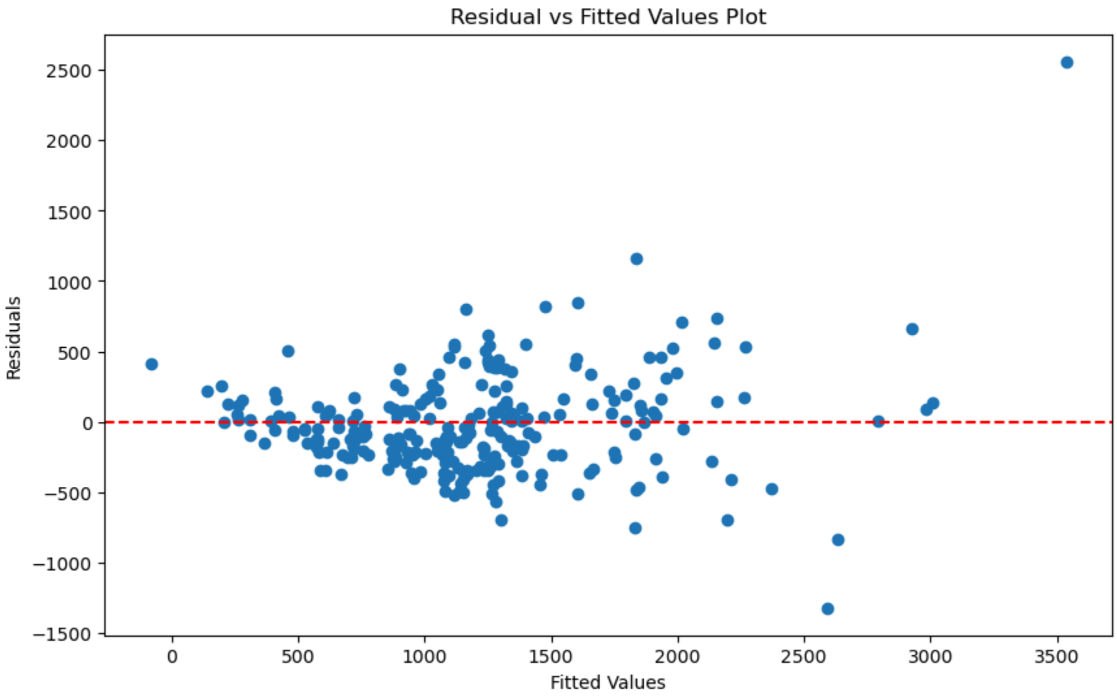
	Actual Laptop Price	Predicted Laptop Price
1179	650.0	605.186930
342	716.0	932.670068
649	1584.0	1530.857782
772	1020.0	939.700915
803	1749.0	1831.412106
...	...	...
701	399.0	612.305397
1105	1413.1	1307.588596
424	2799.0	2266.791672
944	1299.0	1031.307508
65	1983.0	1793.509666

255 rows × 2 columns

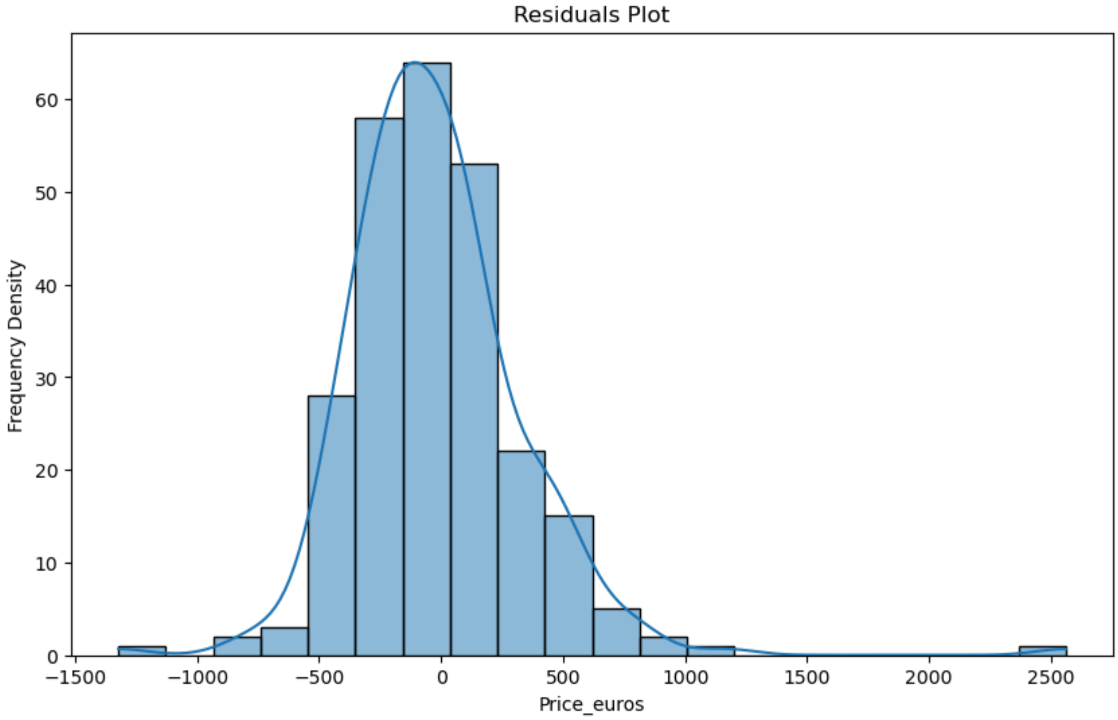
In [59]:
```
## Plot the Predicted vs Actual Prices
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual Prices")
plt.show()
```

Regression diagnostics- Residual plot analysis

In [60]:
```
residuals = y_test-y_pred

# Create a residual vs fitted values plot
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals)
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residual vs Fitted Values Plot')
plt.axhline(y=0, color='red', linestyle='--')
plt.show()
```



In [61]:
```
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True,bins=20)
plt.ylabel('Frequency Density')
plt.title('Residuals Plot')
plt.show()
```



In [62]:
```
from scipy.stats import shapiro

# Perform Shapiro-Wilk test for normality of residuals
residuals = y_test-y_pred
w_stat, p_value = shapiro(residuals)
print(f"Shapiro-Wilk statistic: {w_stat:.4f}")
print(f"p-value: {p_value:.15f}")


if p_value < 0.05:
    print("Reject the null hypothesis. The residuals are not normally distr
else:
    print("Fail to reject the null hypothesis. The residuals are normally d
```

```
Shapiro-Wilk statistic: 0.9006
p-value: 0.000000000006189
Reject the null hypothesis. The residuals are not normally distributed.
```

Random Forest Regressor Model

In [63]:
```
## Create Random Forest Regressor Model
rf_model=RandomForestRegressor(n_estimators=100,random_state=42)

## Train the Model
rf_model.fit(X_train_scaled, y_train)
```

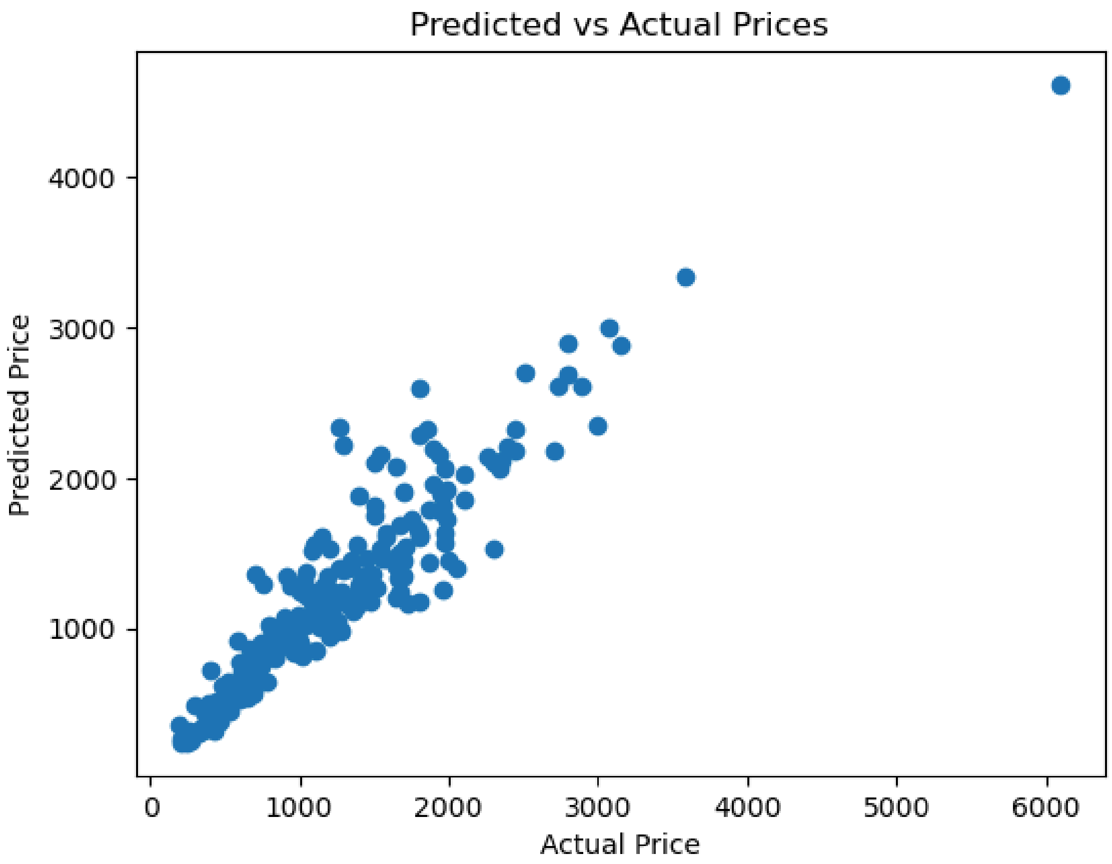
Out[63]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [64]:
```
## Make Prediction on the test Dataset
Prediction=rf_model.predict(X_test_scaled)
```

In [65]:
```
## Evaluate the Model Performance
mae = mean_absolute_error(y_test, Prediction)
mse = mean_squared_error(y_test, Prediction)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, Prediction)
print(f"Mean Absolute Error :{mae:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Absolute Error :161.39
Root Mean Squared Error: 251.75
R-squared: 0.87
```

In [66]:
```
## Plot the Predicted vs Actual prices
plt.scatter(y_test, Prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual Prices")
plt.show()
```



XGboost Algotithm

In [67]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\sagnik samanta\anacond
a3\lib\site-packages (2.1.3)
Requirement already satisfied: numpy in c:\users\sagnik samanta\anaconda3
\lib\site-packages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\users\sagnik samanta\anaconda3
\lib\site-packages (from xgboost) (1.11.1)
Note: you may need to restart the kernel to use updated packages.
```

In [68]:
```
## Import Necessary Library
import xgboost as xgb
```

In [69]:
```
## Create an XGBoost Regressor Model
model = xgb.XGBRegressor(objective='reg:squarederror', max_depth=5, learnin
```

In [70]:
```
## Train the Model
model.fit(X_train_scaled, y_train)
```

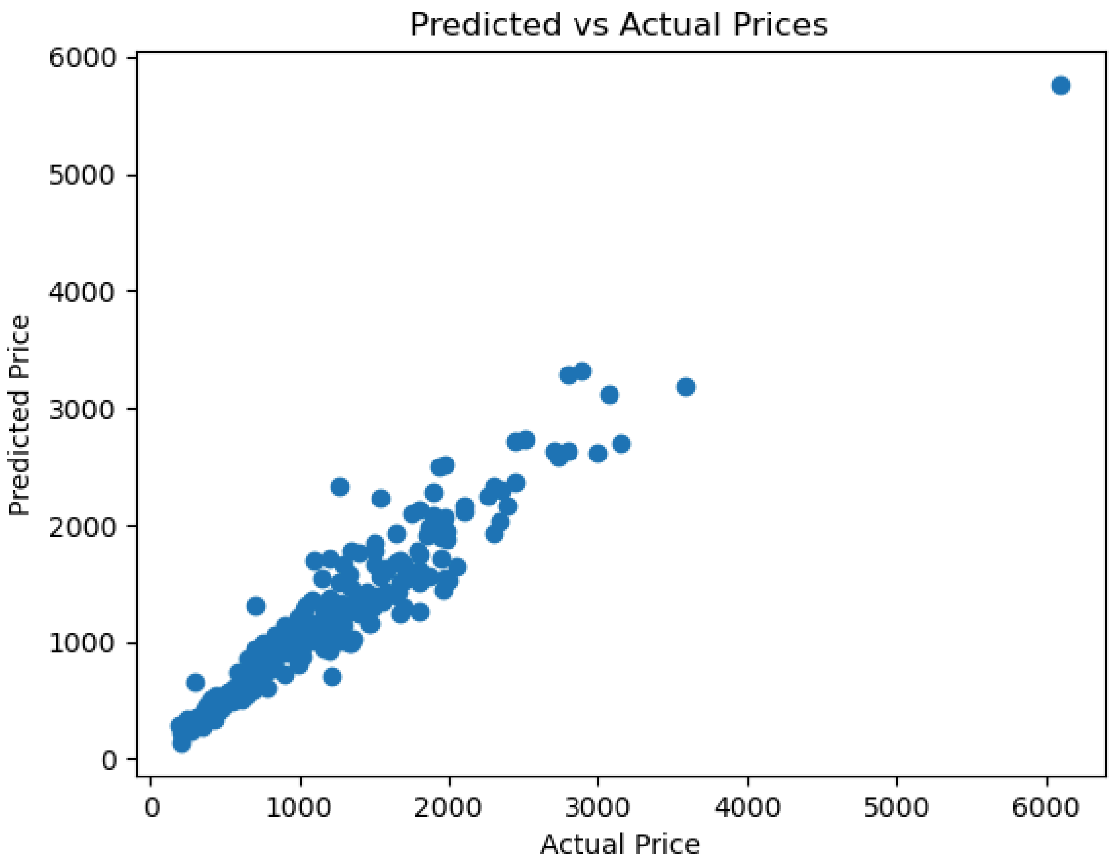
Out[70]:
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_round
s=None,
             enable_categorical=False, eval_metric=None, feature_type
s=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin
=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
```

In [71]:
```
## Make predictions on the test dataset
y_prediction = model.predict(X_test_scaled)
```

In [72]:
```
## Evaluate the Model Performance
mse = mean_squared_error(y_test, y_prediction)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_prediction)
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Root Mean Squared Error: 207.79
R-squared: 0.91
```

In [73]:
```
## Plot the Predicted vs Actual Prices
plt.scatter(y_test, y_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual Prices")
plt.show()
```



Conclusion :

In our Project Multiple Linear Regression, Random Forest Regressor and XGboost have been employed to predict prices of Laptops.By compairing those three models we can see that under XGboost Root Mean Squared Error is 207.79 and Adjusted R-Squared is 0.91.It is evident that XGBoost model exhibits the smallest Root Mean Squared Error and highest Adjusted R-Squared compared to other models.This suggests that XGBoost model provides the highest accuracy and best fit for predicting the prices of Laptops.

In [ ]: