In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn import svm
```

In [2]:

```
df = pd.read_csv('/Users/aminazimi/Downloads/vgsales.csv', encoding = 'utf-8')
df = df.iloc[:5000]
df
```

Out[2]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
...	...	...	...	...	...	...	...	...	...	...	...
4995	4997	Peppa Pig: The Game	Wii	2009.0	Misc	Pinnacle	0.00	0.35	0.00	0.03	0.38
4996	4998	God Eater 2: Rage Burst	PSV	2015.0	Role-Playing	Namco Bandai Games	0.00	0.01	0.37	0.00	0.38
4997	4999	Get Fit with Mel B	PS3	2010.0	Sports	Black Bean Games	0.15	0.17	0.00	0.07	0.38
4998	5000	The Cat in the Hat	GBA	2005.0	Platform	Jack of All Games	0.27	0.10	0.00	0.01	0.38
4999	5001	Naruto Shippuden: Ultimate Ninja Heroes 3	PSP	2009.0	Fighting	Namco Bandai Games	0.13	0.04	0.19	0.03	0.38

5000 rows × 11 columns

In [3]:
```
pd.crosstab(index=df["Publisher"], columns="count")
```

Out[3]:

col_0	count
Publisher	
20th Century Fox Video Games	1
3DO	9
505 Games	31
989 Studios	12
ASC Games	2
...	...
Xplosiv	1
Xseed Games	1
Zoo Digital Publishing	4
Zoo Games	2
mixi, Inc	1

194 rows × 1 columns

In [4]:
```
#pip install patsy
```

In [5]:
```
import patsy
publisher = patsy.dmatrix('C(Publisher)', df, return_type='dataframe')
publisher
#I want to compare the difference between the different publishers
#what effect that makes on the sales of their video games
```

Out[5]:

	Intercept	C(Publisher) [T.3DO]	C(Publisher) [T.505 Games]	C(Publisher) [T.989 Studios]	C(Publisher) [T.ASC Games]	C(Publisher) [T.ASCII Entertainment]	C(Publisher) [T.Acclaim Entertainment]	C(Publisher) [T.Accolade]	C(Publisher) [T.Activision]	C [T
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	Intercept	C(Publisher) [T.3DO]	C(Publisher) [T.505 Games]	C(Publisher) [T.989 Studios]	C(Publisher) [T.ASC Games]	C(Publisher) [T.ASCII Entertainment]	C(Publisher) [T.Acclaim Entertainment]	C(Publisher) [T.Accolade]	C(Publisher) [T.Activision]	C [T
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
4995	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4996	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4997	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4998	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4999	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

4990 rows × 194 columns

In [6]:

```
df = pd.concat([df, publisher], axis=1)
df
```

Out[6]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	...	C(Publisher) [T.Wanadoo]	C(P [In Entert
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	...	0.0	
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	...	0.0	
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	...	0.0	
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	...	0.0	

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	...	C(Publisher)[T.Wanadoo]	C(P [In Entert
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	4997	Peppa Pig: The Game	Wii	2009.0	Misc	Pinnacle	0.00	0.35	0.00	0.03	...	0.0	
4996	4998	God Eater 2: Rage Burst	PSV	2015.0	Role-Playing	Namco Bandai Games	0.00	0.01	0.37	0.00	...	0.0	
4997	4999	Get Fit with Mel B	PS3	2010.0	Sports	Black Bean Games	0.15	0.17	0.00	0.07	...	0.0	
4998	5000	The Cat in the Hat	GBA	2005.0	Platform	Jack of All Games	0.27	0.10	0.00	0.01	...	0.0	
4999	5001	Naruto Shippuden: Ultimate Ninja Heroes 3	PSP	2009.0	Fighting	Namco Bandai Games	0.13	0.04	0.19	0.03	...	0.0	

5000 rows × 205 columns

In [7]:

```
#this is the final table that includes all categorical values of publishers as well as the original dataset
```

In [8]:

```
import statsmodels.api as sm

# Dropping missing data if any
df = df.dropna()

# Creating a simple model with one a constant, one numerical, and four dummies (our categorial predictor)

y = df['Global_Sales']
```

```python
X = df[['NA_Sales', 'C(Publisher)[T.Activision]', 'C(Publisher)[T.Nintendo]','C(Publisher)[T.505 Games]', 'C(Pu
X = sm.add_constant(X)
model = sm.OLS(y, X).fit() # Note that I keep changing the model names
print(model.summary())

# Always remember what your COMPARISON gorup is (here it's "at_home")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Global_Sales   R-squared:                       0.873
Model:                            OLS   Adj. R-squared:                  0.873
Method:                 Least Squares   F-statistic:                     6782.
Date:                Sun, 11 Sep 2022   Prob (F-statistic):               0.00
Time:                        17:33:14   Log-Likelihood:                -6610.9
No. Observations:                4924   AIC:                         1.323e+04
Df Residuals:                    4918   BIC:                         1.327e+04
Df Model:                           5
Covariance Type:            nonrobust
=====================================================================================================
                                        coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
const                                 0.1834      0.016     11.571      0.000       0.152       0.214
NA_Sales                              1.7347      0.010    176.611      0.000       1.715       1.754
C(Publisher)[T.Activision]           -0.1793      0.049     -3.686      0.000      -0.275      -0.084
C(Publisher)[T.Nintendo]              0.5329      0.045     11.733      0.000       0.444       0.622
C(Publisher)[T.505 Games]            -0.1228      0.167     -0.735      0.462      -0.451       0.205
C(Publisher)[T.Activision Blizzard]   0.1578      0.927      0.170      0.865      -1.660       1.976
==============================================================================
Omnibus:                     2379.338   Durbin-Watson:                   1.921
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1033089.505
Skew:                           0.988   Prob(JB):                         0.00
Kurtosis:                      73.933   Cond. No.                         117.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/Users/aminazimi/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a
future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [9]:

```python
#two categorical values are significant enough (p<0.05) which are NA_Sales and C(Publisher)[T.Nintendo]
#they have a decent effect on global sales
#NA_Sales (North America Sales) makes up the majority of the prediction for global sales --> some multicollinea
#Nintendo is probably one of the best game publishers with high sales around the world
```

In [10]:

```
from sklearn.model_selection import GridSearchCV
```

In [11]:

```
df = df.sample(1000)
y = df['Global_Sales']
X = df[['NA_Sales', 'C(Publisher)[T.Activision]', 'C(Publisher)[T.Nintendo]','C(Publisher)[T.505 Games]', 'C(Pu
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.4, random_state = 0)

param_grid = [{'kernel': ['poly', 'linear', 'rbf'], 'C': [1, 10, 20, 50]}]
svr = svm.SVR(kernel='rbf', C = 20)
```

In [12]:

```
grid_search = GridSearchCV(svr, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Out[12]:

```
GridSearchCV(cv=5, estimator=SVR(C=20),
             param_grid=[{'C': [1, 10, 20, 50],
                          'kernel': ['poly', 'linear', 'rbf']}])
```

In [13]:

```
grid_search.best_params_
```

Out[13]:

```
{'C': 10, 'kernel': 'linear'}
```

In [14]:

```
shap_model = grid_search.best_estimator_
shap_model
```

Out[14]:

```
SVR(C=10, kernel='linear')
```

In [15]:

```
#grid search CV using Support Vector Regression (because we are predicting a number, not a category)
```

In [ ]:

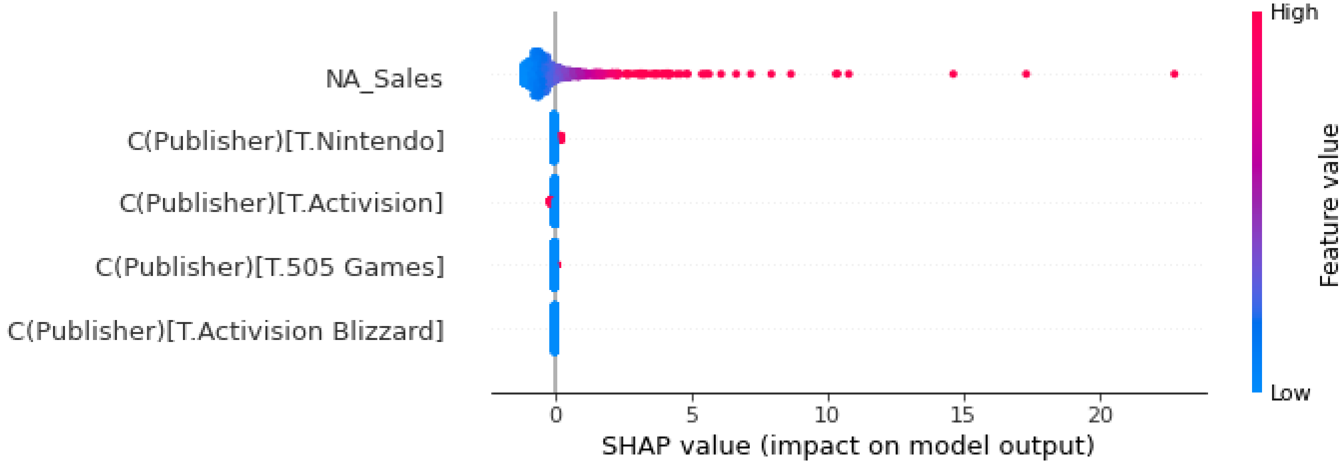
```
#conda install -c conda-forge shap
```

In [19]:

```
import shap
shap.initjs()
explainer = shap.KernelExplainer(shap_model.predict, X)
shap_values = explainer.shap_values(X)
```

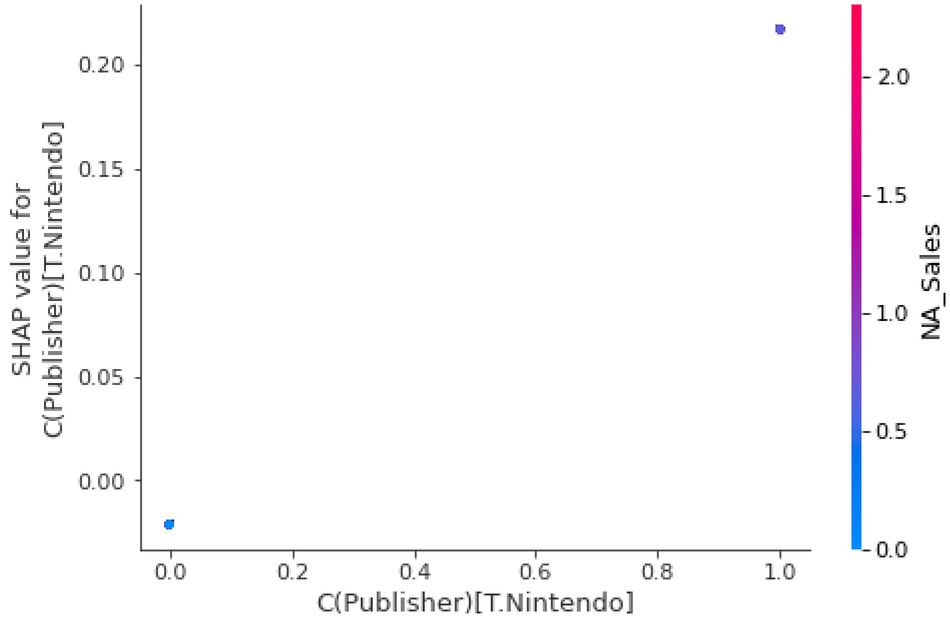
```
Using 1000 background data samples could cause slower run times. Consider using shap.sample(data, K) or shap.km
eans(data, K) to summarize the background as K samples.
```

In [20]:
```
shap.summary_plot(shap_values, X)
```



In [21]:
```
#NA_Sales has the biggest impact
#the sales terrotories are split into 4 groups: NA, EU, JP, other
#NA has the largest video game sales, which would be a strong indicator of total global sales
#this has some multicollinearity to it
#i wanted to analyze the sales of a game based on its publishers, and chose 4 of the biggest ones, but it didn'
```

In [22]:
```
shap.dependence_plot("C(Publisher)[T.Nintendo]", shap_values, X)
```



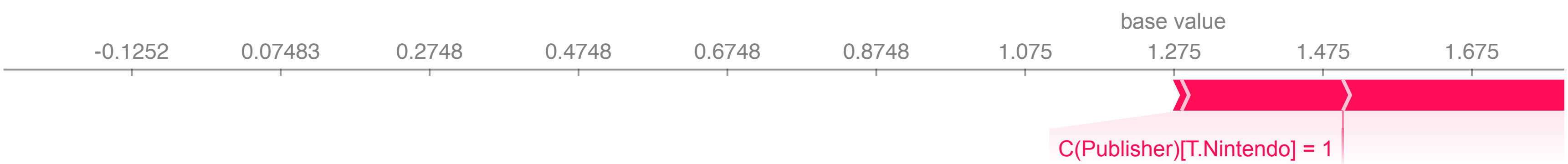
In [23]:

```
#if the publisher is Nintendo, the SHAP value increases by a small amount
#this means that consumers of video games aren't looking for Nintendo games specifically,
#but it could be a positive factor in purchasing if they are on the edge
```

In [24]:

```
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```

Out[24]:



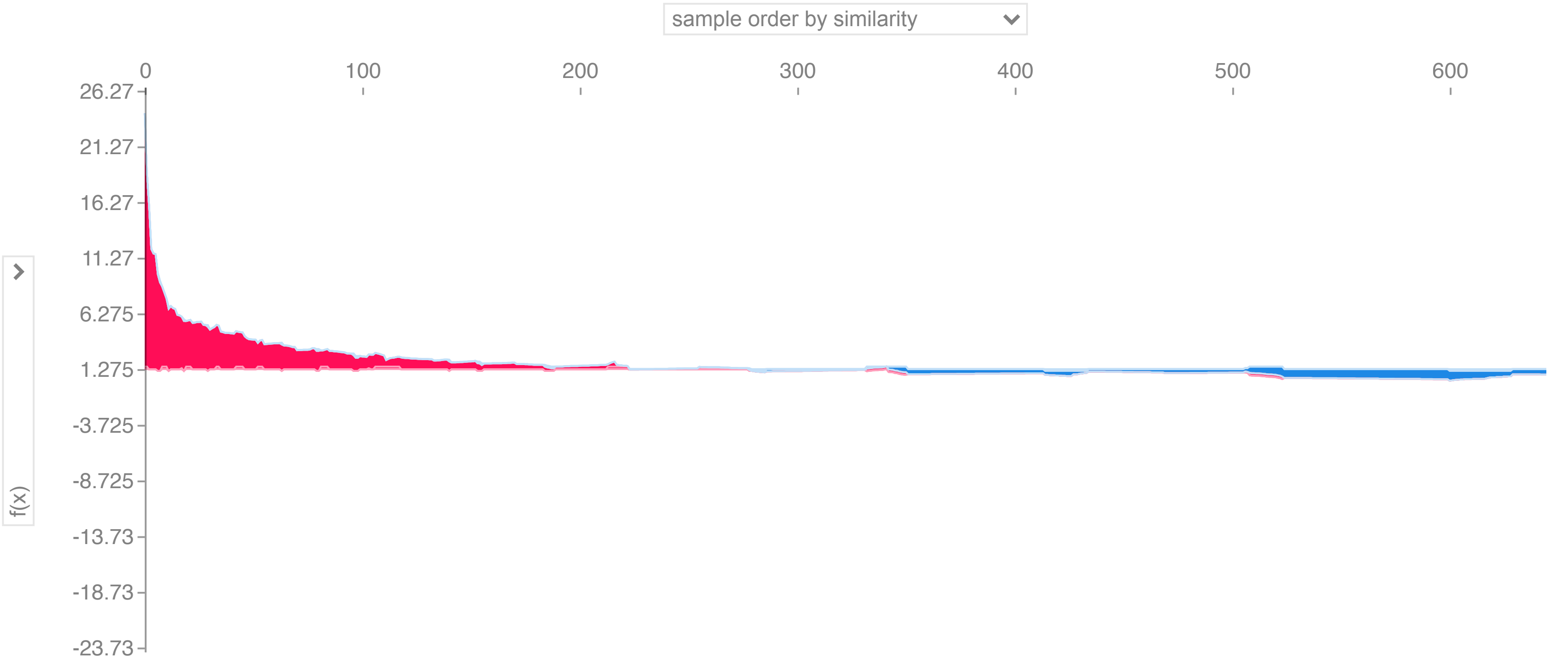
In [25]:

```
#NA_Sales is the siginificant predictor that is predicting the probability of global sales
#probability of 0.88 of predicting global sales
```

In [26]:

```
shap.force_plot(explainer.expected_value, shap_values, X)
```

Out[26]:



In [27]:

```
#this graph can be changed to fit someone's needs for visualizing these relationships
#we can see in this graph that NA_Sales is dominating output value just because of how big a predictor it is fo
#we can explore other relationships between predictors and outcomes, etc.
```

In [ ]: