In [1]:

```
import pandas as pd
```

In [9]:

```
#Create list of tickers that openinsider won't recognize (generally foreign com
#that don't have to report insider transactions, but in some cases firms were e
#merged into others, acquired, or went bankrupt)
#remove_these_firms = ['ADAPT:ST','0A1R.IL','MCG.JO','CLEGF','RVRA','OTC:EPOKY'
 #                     'MFGP','MTAGF','IDRSF','ETTYF','TSX:IPCO','ECN.TO','UN01'
  #                    'HHDS','RACE','TSSLF','AAMMF','S32.AX','LSE:INDV','LBDRK'
   #                   'LTRPB','PGN (BANKRUPT)','HEL:VALMT','ASX:ORA','Munich:OS
    #                  'Paris:FNAC','CHRYY','POOSF','NTCPF','ENRFF','TSRYY','APE
```

In [10]:

```
#remove tickers from original list that openinsider won't recognize
#tickers = [elem for elem in tickers if elem not in remove_these_firms]
```

In [11]:

```
#create empty dataframe that will later be populated by the pulled transactions
#insider = pd.DataFrame()
```

In [12]:

```
#loop through companies list and get the last 500 buy or sell transactions#
#for ticker in tickers:
#    insider1 = pd.read_html(f'http://openinsider.com/screener?s={ticker}&o=&pl
#    #change the object that we pull the data from, from the entire page at the
#    insider1 = insider1[-3]
#   insider = pd.concat([insider,insider1])
```

In [13]:

```
#Removing columns that aren't useful
#insider = insider.drop([0,1,'1d','1m','1w','6m'],axis=1)
```

In [14]:

```
#Confirming that colums were removed
#insider.head()
```

In [15]:

```
#Defining own column names due to difficulty removing filing date and delta (sy
#insider.columns = ['Filing Date','Name','Owned','Price','Qty','Ticker','Title'
```

In [16]:

```
#Drop unnecessary features
#insider = insider.drop(['Filing Date','Name','Owned','Title','X','Delta Owned'
```

In [17]:

```
#insider.head()
```

In [18]:

```
#Convert value in string format to value in float format
#insider['Value No $'] = insider['Value'].str.replace('$','')
#insider['Value No $ or Comma'] = insider['Value No $'].str.replace(',','')
#insider['Value No $ or Comma'] = insider['Value No $ or Comma'].astype(float)
```

In [19]:

```
#Clean redundant/transition columns used to convert from string to float
#insider = insider.drop(['Value','Value No $'],axis=1)
```

In [20]:

```
#insider.columns = ['Price','Qty','Ticker','Trade Date','Trade Type','Value']
```

In [21]:

```
#Sum values based on ticker
#pd.set_option('display.max_rows', None)
#by_ticker = insider.groupby('Ticker')['Value'].sum()
#generate CSV so I can copy-paste these values into the Excel dataset and use 1
#by_ticker.to_csv('value_by_ticker.csv')
```

In [2]:

```
#import dataset for algo
real_deal_df = pd.read_csv('Spinoff Situations Full Data.csv')
```

In [3]:

```
real_deal_df.head()
```

Out[3]:

	Unnamed: 0	Spinoff_Name	Spinoff_Ticker	Sector	P0	Year_Return_
0	0	Adapteo	ADAPT:ST	Real Estate	11.99655	
1	1	IAA	IAA	Industrials	43.68000	
2	2	Corteva	CTVA	Materials	25.43000	
3	3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	
4	4	Alcon	ALC	Healthcare	52.54000	

In [4]:

```
#remove unnecessary features
real_deal_df = real_deal_df.drop('Unnamed: 0',axis=1)
```

In [5]:

```
#Check that features are gone
real_deal_df.head()
```

Out[5]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [95]:

```
#real_deal_df.columns = ['Spinoff_Name','Spinoff_Ticker','Sector','P0','1-Year_
```

In [96]:

```
#real_deal_df.head()
```

In [6]:

```
import numpy as np
```

In [7]:

```
real_deal_df.head()
```

Out[7]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [99]:

```
real_deal_df['Sector'].value_counts()
```

Out[99]:

```
Consumer Discretionary    29
Energy                    23
Industrials               22
Real Estate               22
Communication Services    19
Materials                 18
Healthcare                16
Financials                13
Information Technology    12
Consumer Staples           2
Utilities                  1
Name: Sector, dtype: int64
```

In [27]:

```
#real_deal_df[real_deal_df['Insiders_Purchase_vs_Sell_Margin_by_Value'].isna()]
```

In [28]:

```
#Determine median value of each secto
#healthcare_median = real_deal_df[real_deal_df['Sector']=='Healthcare']['Inside
```

In [29]:

```
#real_estate_median = real_deal_df[real_deal_df['Sector']=='Real Estate']['Insi
```

In [30]:

```
#energy_median = real_deal_df[real_deal_df['Sector']=='Energy']['Insiders_Purch
```

In [31]:

```
#comms_services_median = real_deal_df[real_deal_df['Sector']=='Communication Se
#comms_services_median
```

In [32]:

```
#industrials_median = real_deal_df[real_deal_df['Sector']=='Industrials']['Insi
```

In [33]:

```
#financials_median = real_deal_df[real_deal_df['Sector']=='Financials']['Inside
```

In [34]:

```
#info_tech_median = real_deal_df[real_deal_df['Sector']=='Information Technolog
```

In [35]:

```
#cons_disc_median = real_deal_df[real_deal_df['Sector']=='Consumer Discretionar
```

In [36]:

```
#cons_staples_median = real_deal_df[real_deal_df['Sector']=='Consumer Staples']
```

In [37]:

```
#materials_median = real_deal_df[real_deal_df['Sector']=='Materials']['Insiders
```

In [38]:

```
#real_deal_df.loc['ADAPT:ST','Insiders Purchase vs Sell Margin by Value'] = rea
#real_estate_median
```

In [39]:

```
#real_deal_df.loc['HHDS','Insiders Purchase vs Sell Margin by Value'] = real_es
```

In [40]:

```
#real_deal_df.loc['FSV','Insiders Purchase vs Sell Margin by Value'] = real_est
```

In [41]:

```
#real_deal_df.loc['BPY','Insiders Purchase vs Sell Margin by Value'] = real_est
```

In [42]:

```
#real_deal_df = real_deal_df.drop('Insiders Purchase vs Sell Margin by Value',a
```

In [43]:

```
#real_deal_df.head()
```

In [44]:

```
#real_deal_df[(real_deal_df['Insiders_Purchase_vs_Sell_Margin_by_Value'].isna()
```

In [45]:

```
#This worked. This is the method to use
#real_deal_df.loc[(real_deal_df['Spinoff_Ticker']=='FSV') | (real_deal_df['Spir
```

In [46]:

```
#real_deal_df[real_deal_df['Sector']=='Real Estate']
```

In [47]:

```
#real_deal_df = real_deal_df.drop(['ADAPT:ST','HHDS','FSV','BPY'])
```

In [48]:

```
#real_deal_df.tail()
```

In [49]:

```
#This works better
#real_deal_df.loc[(real_deal_df['Sector']=='Communication Services') & (real_de
```

In [50]:

```
#real_deal_df[real_deal_df['Sector']=='Communication Services']
```

In [51]:

```
#Checked to see how much data is still missing
#real_deal_df.isna().sum()
```

In [52]:

```
#real_deal_df.loc[(real_deal_df['Sector']=='Real Estate') & (real_deal_df['Insi
```

In [53]:

```
#real_deal_df[real_deal_df['Sector']=='Real Estate']
```

In [54]:

```
#Impute remaining sectors' missing values
#real_deal_df.loc[(real_deal_df['Sector']=='Energy') & (real_deal_df['Insiders_
#real_deal_df.loc[(real_deal_df['Sector']=='Materials') & (real_deal_df['Inside
#real_deal_df.loc[(real_deal_df['Sector']=='Industrials') & (real_deal_df['Insi
#Used healthcare median for utilities because there was only one utility firm i
#real_deal_df.loc[(real_deal_df['Sector']=='Utilities') & (real_deal_df['Inside
#real_deal_df.loc[(real_deal_df['Sector']=='Healthcare') & (real_deal_df['Insic
#real_deal_df.loc[(real_deal_df['Sector']=='Financials') & (real_deal_df['Insic
#real_deal_df.loc[(real_deal_df['Sector']=='Consumer Discretionary') & (real_de
#real_deal_df.loc[(real_deal_df['Sector']=='Consumer Staples') & (real_deal_df[
#real_deal_df.loc[(real_deal_df['Sector']=='Information Technology') & (real_de
```

In [55]:

```
#Check for missing data again
#real_deal_df.isna().sum()
```

In [56]:

```
#real_deal_df[real_deal_df['P0'].isna()]
```

In [57]:

```
#Create variable for energy median P0
#energy_P0_median = real_deal_df[real_deal_df['Sector']=='Energy']['P0'].median
```

In [58]:

```
#Impute missing P0 values. All were in the energy sector, so used the energy me
#real_deal_df['P0'] = real_deal_df['P0'].fillna(energy_P0_median)
```

In [59]:

```
#No more missing values
#real_deal_df.isna().sum()
```

In [60]:

```
#real_deal_df
```

In [61]:

```
#Create new, consolidated CSV file
#real_deal_df.to_csv('Spinoff Situations Cons.csv')
```

In [8]:

```
#find 75th, 50th, and 25th percentiles for creating low/mid/high P0 categorical
q75, q50, q25 = np.percentile(real_deal_df['P0'],[75,50,25])
```

In [9]:

```
#Find dispersion of P0 data to determine if it's worth it to
#create a dummy variable based on statistical distribution
#real_deal_df[(real_deal_df['P0'] > q25) & (real_deal_df['P0'] < q75)]['P0'].cc
```

In [10]:

```
#real_deal_df[(real_deal_df['P0'] < q25) | (real_deal_df['P0'] > q75)]['P0'].cc
```

In [11]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [10]:

```
#qt = QuantileTransformer(n_quantiles=100,output_distribution='normal')
```

In [11]:

```
#Define function to calculate standard normal distribution
#To create distribution of P0 that can be visualized easily
#Raw data distribution ranges from .04 to 33,000
#mu = sum(real_deal_df['P0']/len(real_deal_df['P0']))
#import statistics as st
#P0_stdev = st.pstdev(real_deal_df['P0'])
#P0_stdev

#def std_norm_transform(x):
 #       st_normed_x = (x - mu) / P0_stdev
  #      return st_normed_x
```

In [12]:

```
#Define function to apply to P0 for splitting into low/medium/high buckets
#def buckets(y):
 #   if y <= q25:
  #      return("Low")
   # elif (y > q25) & (y <= q75):
    #    return("Mid")
    #else:
     #   return("High")
```

In [13]:

```
#Tried normalizing the P0 data because it had extremely high dispersion that sc
#real_deal_df['Std Norm Result'] = real_deal_df['P0'].apply(lambda x:std_norm_t
```

In [70]:

```
#real_deal_df.head()
```

In [115]:

```
#Create the buckets column
#real_deal_df['P0 Bucket'] = real_deal_df['P0'].apply(lambda x:buckets(x))
```

In [77]:

```
#Check that buckets populated
#real_deal_df.head()
```

In [71]:

```
#second_deal_df = real_deal_df[['Spinoff_Name','Spinoff_Ticker','Sector','Std N
```

In [72]:

```
#second_deal_df.head()
```

In [73]:

```
#sns.distplot(second_deal_df['Std Norm Result'])
```

In [78]:

```
#real_deal_df[real_deal_df['P0'] == real_deal_df['P0'].max()]
```

In [12]:

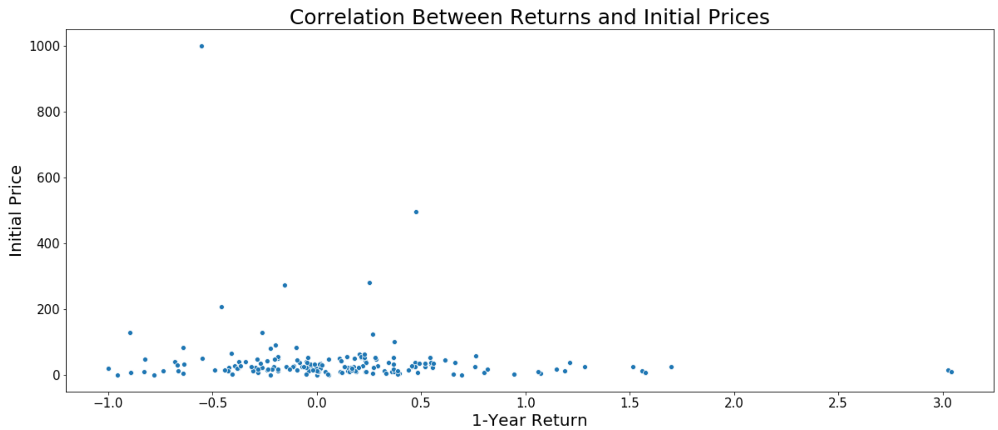
```
#Drop Phio as the outlier messing up the visuals
real_deal_df = real_deal_df.drop([154])
```

In [8]:

```
#Visually gauge relationship between 1-year return and initial price.
#Very weak, but want some concrete numbers to quantify this
plt.figure(figsize=(20,8))
sns.scatterplot(real_deal_df['1-Year_Return_As_Decimal'],real_deal_df['P0'])
plt.title('Correlation Between Returns and Initial Prices',fontsize=25)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel(fontsize=20,s="1-Year Return")
plt.ylabel(fontsize=20,s='Initial Price')
```

Out[8]:

```
Text(0,0.5,'Initial Price')
```

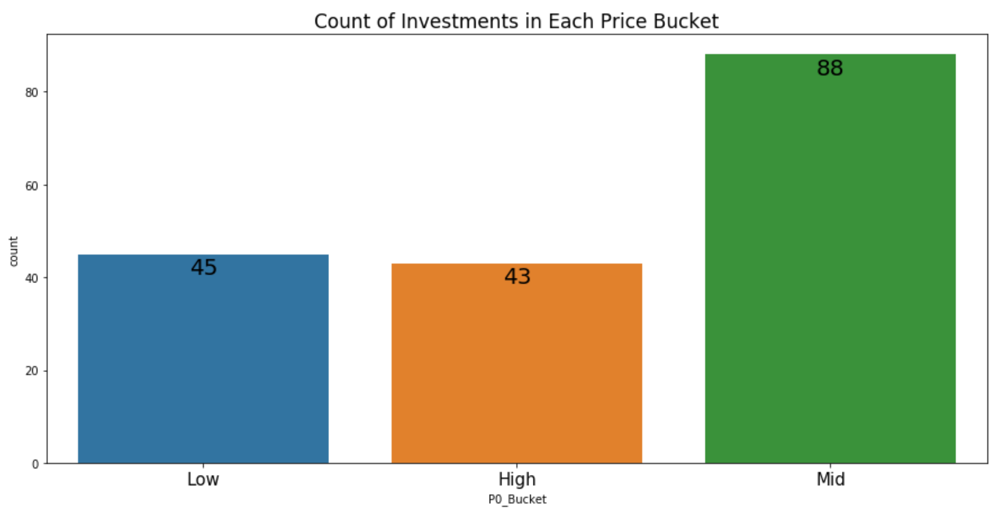


In [9]:

```
#Visually gauge dataset balance between buckets
plt.figure(figsize=(15,7))
plt.title('Count of Investments in Each Price Bucket',fontsize=17)
plt.xticks(fontsize=13)
bucket_plot = sns.countplot(real_deal_df['P0_Bucket'])
plt.xticks(fontsize=15)
for each in bucket_plot.patches:
    bucket_plot.annotate(format(each.get_height()),
                         (each.get_x() + each.get_width() / 2,
                         each.get_height()),ha='center',va='center',
                         size=20,xytext=(0,-12),
                         textcoords='offset points')
bucket_plot
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f0324d278>
```

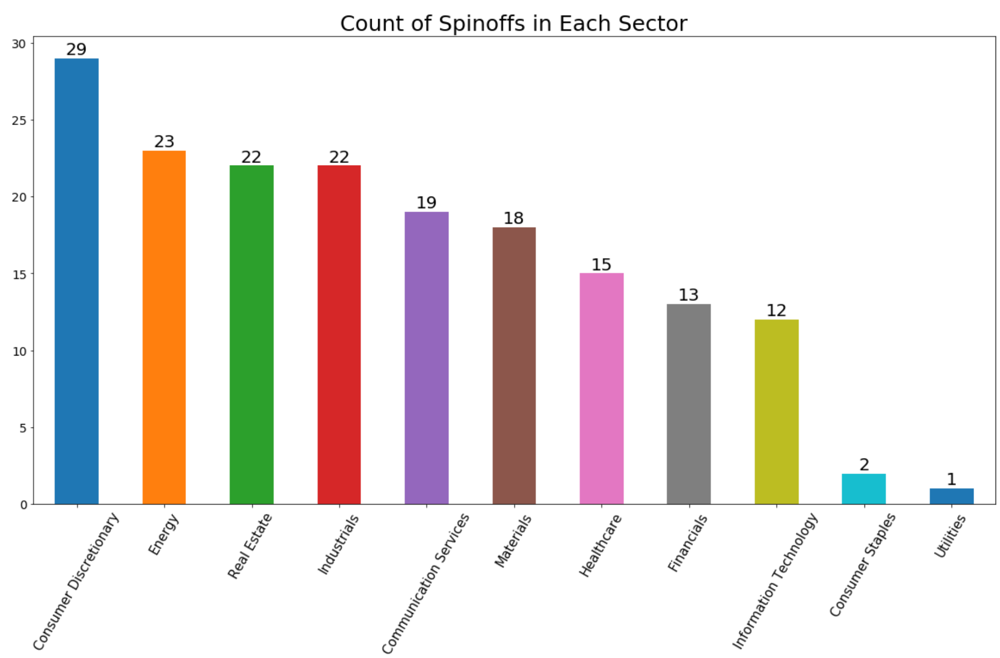


In [10]:

```
#order these from greatest to least
plt.figure(figsize=(20,10))
plt.title('Count of Spinoffs in Each Sector',fontsize=25)
plots = real_deal_df['Sector'].value_counts().plot(kind="bar")
plt.xticks(fontsize=15,rotation=60)
plt.yticks(fontsize=14)
for bar in plots.patches:
    plots.annotate(format(bar.get_height()),
                   (bar.get_x() + bar.get_width() / 2,
                   bar.get_height()),ha='center',va='center',
                   size=20, xytext=(0,10),
                   textcoords='offset points')
plots
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f02787be0>
```



In [25]:

```
#Very weak correlation between initial price and return
real_deal_df[['1-Year_Return_As_Decimal','P0']].corr()
```

Out[25]:

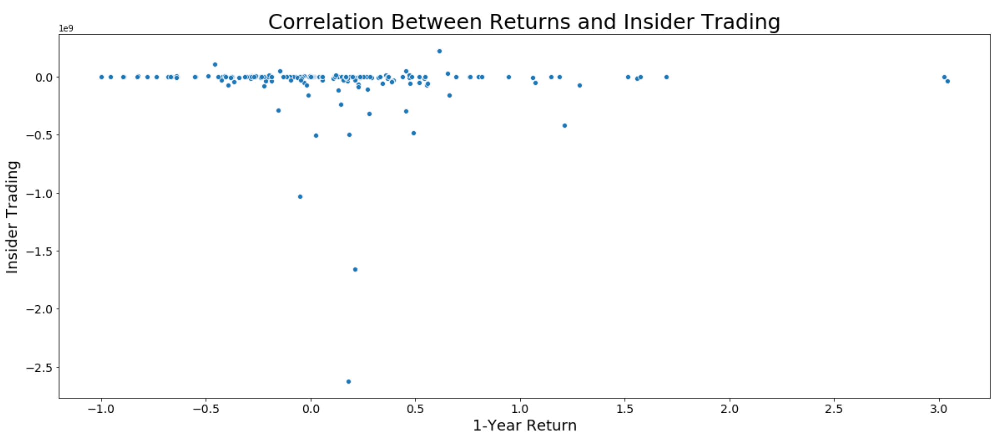
	1-Year_Return_As_Decimal	P0
1-Year_Return_As_Decimal	1.000000	-0.096235
P0	-0.096235	1.000000

In [90]:

```
plt.figure(figsize=(20,8))
sns.scatterplot(real_deal_df['1-Year_Return_As_Decimal'],real_deal_df['Insiders
plt.title('Correlation Between Returns and Insider Trading',fontsize=25)
plt.xlabel(s='1-Year Return',fontsize=18)
plt.ylabel('Insider Trading',fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
```

Out[90]:

```
(array([-3.0e+09, -2.5e+09, -2.0e+09, -1.5e+09, -1.0e+09, -5.0e+
08,
         0.0e+00,  5.0e+08]), <a list of 8 Text yticklabel objec
ts>)
```



In [109]:

```
#Also a very weak correlation.
real_deal_df[['1-Year_Return_As_Decimal','Insiders_Buy_Sell_Margin_by_Value']].
```

Out[109]:

	1-Year_Return_As_Decimal	Insiders_Buy_Sell_M
1-Year_Return_As_Decimal	1.000000	
Insiders_Buy_Sell_Margin_by_Value	-0.046893	

In [110]:

```
real_deal_df.head()
```

Out[110]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [111]:

```
#So we know now that the initial price and insider trading margin are poor driv
#Still going to run a logistic regression model to see if the sector tells us a
#very little for predicting spinoff investment success
```

In [26]:

```
#Create dummy variable for sector
sector_dummy = pd.get_dummies(real_deal_df['Sector'],drop_first=True)
```

In [27]:

```
#confirm dummy was created correctly
sector_dummy.head()
```

Out[27]:

	Consumer Discretionary	Consumer Staples	Energy	Financials	Healthcare	Industrials	Informat Technol
0	0	0	0	0	0	0	
1	0	0	0	0	0	1	
2	0	0	0	0	0	0	
3	1	0	0	0	0	0	
4	0	0	0	0	1	0	

In [28]:

```
#Concat initial df and dummy df
real_deal_df = pd.concat([real_deal_df,sector_dummy],axis=1)
```

In [29]:

```
#See if combined df looks right
real_deal_df.head()
```

Out[29]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [30]:

```
#Remove Unnamed feature
#real_deal_df = real_deal_df.drop("Unnamed: 0",axis=1)
```

In [31]:

```
new_df = real_deal_df
```

In [32]:

```
new_df.head()
```

Out[32]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [33]:

```
#Define function to evaluate good or bad investments based on return
def good_investment(z):
    if z > .1:
        return("Good")
    else:
        return("Bad")
```

In [34]:

```
new_df['Decision'] = new_df['1-Year_Return_As_Decimal'].apply(lambda x:good_inv
```

In [35]:

```
decision_dummy = pd.get_dummies(new_df,columns=['Decision'],drop_first=True)
```

In [36]:

```
new_df = decision_dummy
```

In [37]:

```
new_df.head()
```

Out[37]:

uy_Sell_Margin_by_Value	P0_Bucket	Consumer Discretionary	Consumer Staples	Energy	Financials
-1181578.5	Low	0	0	0	0
-22050.0	High	0	0	0	0
401419.0	Mid	0	0	0	0
-490071.0	Mid	1	0	0	0
19723.0	High	0	0	0	0

In [20]:

```
from sklearn.model_selection import train_test_split
```

In [45]:

```
#new_df = new_df.drop(['Spinoff_Name','Spinoff_Ticker','Sector','P0_Bucket'],ax
```

In [57]:

```
new_df.head()
new_df.to_csv('Dummified Spinoff Data.csv')
```

In [38]:

```
X = new_df[['P0','Insiders_Buy_Sell_Margin_by_Value','Consumer Discretionary','
y = new_df['Decision_Good']
```

In [65]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rando
```

In [66]:

```
from sklearn.linear_model import LogisticRegression
```

In [67]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

Out[67]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_int
ercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', t
ol=0.0001,
          verbose=0, warm_start=False)
```

In [68]:

```
#Would not converge.
import statsmodels.api as sm
log_reg = sm.Logit(y_train, X_train).fit()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 63.294422
         Iterations: 35
```

```
C:\Users\magilmartin\AppData\Local\Continuum\anaconda3.1\lib\sit
e-packages\statsmodels\base\model.py:508: ConvergenceWarning: Ma
ximum Likelihood optimization failed to converge. Check mle_retv
als
  "Check mle_retvals", ConvergenceWarning)
```

In [69]:

```
predictions = logmodel.predict(X_test)
```

In [70]:

```
from sklearn.metrics import classification_report
```

In [71]:

```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.74      0.52      0.61        33
          1       0.56      0.77      0.65        26

avg / total       0.66      0.63      0.62        59
```

In [72]:

```
print(logmodel.coef_, logmodel.intercept_)
```

```
[[-1.15942857e-14 -5.59291712e-09 -6.01881677e-17  1.37904550e-1
7
  -7.39954140e-17 -8.46336605e-17  3.37009370e-17 -2.67623275e-1
7
  -2.38132825e-18  1.32953008e-17 -1.70592669e-17  1.05565451e-1
7]] [-1.16183251e-16]
```

In [73]:

```
print(log_reg.summary())
```

```
                           Logit Regression Results
================================================================
==============
Dep. Variable:          Decision_Good   No. Observations:
117
Model:                          Logit   Df Residuals:
105
Method:                           MLE   Df Model:
11
Date:                Sun, 12 Sep 2021   Pseudo R-squ.:
inf
Time:                        21:41:03   Log-Likelihood:
-7405.4
converged:                      False   LL-Null:
0.0000
                                        LLR p-value:
1.000
================================================================
=====================================
                                        coef    std err
z      P>|z|      [0.025      0.975]
----------------------------------------------------------------
-------------------------------------
P0                                    0.0002      0.002      0.1
11      0.911      -0.004       0.004
Insiders_Buy_Sell_Margin_by_Value -6.394e-09   3.01e-09     -2.1
22      0.034   -1.23e-08   -4.88e-10
Consumer Discretionary               -0.4419      0.463     -0.9
55      0.340      -1.349       0.465
Consumer Staples                     16.9542   5093.192      0.0
03      0.997   -9965.519    9999.427
Energy                               -0.7301      0.551     -1.3
24      0.185      -1.811       0.350
Financials                           -1.8039      0.988     -1.8
26      0.068      -3.740       0.133
Healthcare                            0.4158      0.652      0.6
37      0.524      -0.863       1.694
Industrials                          -0.2741      0.549     -0.4
99      0.617      -1.350       0.802
Information Technology               -0.1664      0.823     -0.2
02      0.840      -1.779       1.446
Materials                             0.2557      0.765      0.3
34      0.738      -1.244       1.756
Real Estate                          -0.1998      0.522     -0.3
```

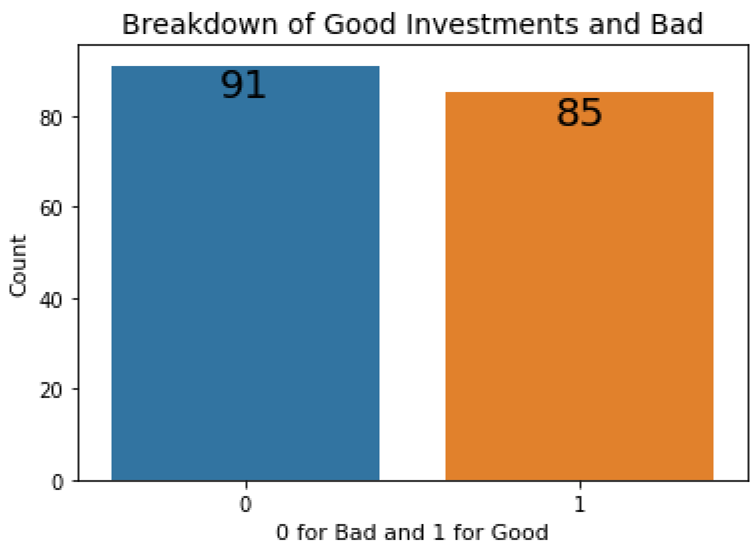
```
83      0.702      -1.223       0.823
Utilities                            14.5654   2111.788      0.0
07      0.994   -4124.464    4153.595
================================================================
=====================================
```

```
C:\Users\magilmartin\AppData\Local\Continuum\anaconda3.1\lib\sit
e-packages\statsmodels\base\model.py:488: HessianInversionWarnin
g: Inverting hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\magilmartin\AppData\Local\Continuum\anaconda3.1\lib\sit
e-packages\statsmodels\base\model.py:488: HessianInversionWarnin
g: Inverting hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\magilmartin\AppData\Local\Continuum\anaconda3.1\lib\sit
e-packages\statsmodels\discrete\discrete_model.py:3313: RuntimeW
arning: divide by zero encountered in double_scalars
  return 1 - self.llf/self.llnull
```

In [82]:

```
#Balance of good investments vs bad
breakdown_plot = sns.countplot(new_df['Decision_Good'])
plt.title('Breakdown of Good Investments and Bad', fontsize=14)
plt.xlabel(s='0 for Bad and 1 for Good',fontsize=11)
plt.ylabel(s='Count',fontsize=11)

for each in breakdown_plot.patches:
    breakdown_plot.annotate(format(each.get_height()),
                   (each.get_x() + each.get_width() / 2,
                   each.get_height()),ha='center',va='center',
                   size=20, xytext=(0,-10),
                   textcoords='offset points')
```



In [143]:

```
real_deal_df.groupby('Sector').describe()
```

Out[143]:

Sector	count	mean	std	min	25%	50%	
Communication Services	19.0	55.705263	107.955961	0.710000	18.9800	28.090	50.0
Consumer Discretionary	29.0	33.499310	28.015087	0.040000	12.1900	24.920	42.9
Consumer Staples	2.0	38.720000	19.063599	25.240000	31.9800	38.720	45.4
Energy	23.0	35.434441	53.965615	5.900000	15.1250	20.900	33.2
Financials	13.0	102.020648	270.295563	3.168418	20.3600	25.240	36.3
Healthcare	15.0	54.008900	72.043535	2.530000	14.1900	27.940	52.3
Industrials	22.0	33.114029	43.579876	1.881743	10.8525	22.695	34.9
Information Technology	12.0	27.972500	24.008847	2.800000	11.0450	19.335	43.2
Materials	18.0	21.921951	17.918109	0.118000	7.1525	22.215	31.7
Real Estate	22.0	26.835752	26.201205	0.220000	12.2300	19.360	33.1
Utilities	1.0	24.590000	NaN	24.590000	24.5900	24.590	24.5

11 rows × 104 columns

In [145]:

```
real_deal_df.head()
```

Out[145]:

	Spinoff_Name	Spinoff_Ticker	Sector	P0	1-Year_Return_As_Decimal
0	Adapteo	ADAPT:ST	Real Estate	11.99655	1.189817
1	IAA	IAA	Industrials	43.68000	0.230540
2	Corteva	CTVA	Materials	25.43000	0.756193
3	Kontoor Brands	KTB	Consumer Discretionary	15.91000	3.027027
4	Alcon	ALC	Healthcare	52.54000	0.366388

In [146]:

```
#Probability that the average spinoff in a given sector will be a good investme
real_deal_df[real_deal_df['Decision']=='Good'].groupby('Sector')['Decision'].cc
```

Out[146]:

```
Sector
Communication Services    0.631579
Consumer Discretionary    0.413793
Consumer Staples          1.000000
Energy                    0.434783
Financials                0.153846
Healthcare                0.600000
Industrials               0.500000
Information Technology    0.583333
Materials                 0.444444
Real Estate               0.500000
Utilities                 1.000000
Name: Decision, dtype: float64
```

In [15]:

```
#Create dictionary to feed dataframe below
dict = {'Communication Services':.631579,'Consumer Discretionary':.413793,'Cons
       'Healthcare':.562500,'Industrials':.5,'Information Technology':.583333,'
```

In [16]:

```
#Create dataframe to show likelihood of good investment in each sector
#with no additional noise
prob_df = pd.DataFrame.from_dict(dict,orient='index')
```

In [17]:

```
prob_df.columns = ['Likelihood_of_Good_Investment']
```

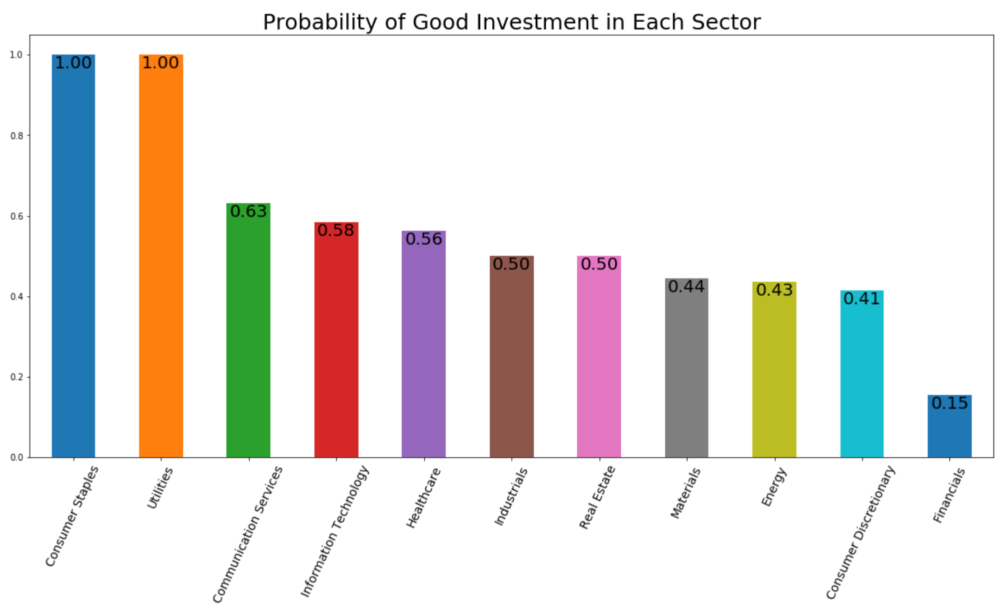
In [18]:

```
prob_df = prob_df.sort_values('Likelihood_of_Good_Investment',ascending=False)
```

In [20]:

```
plt.figure(figsize=(20,9))
plt.title('Probability of Good Investment in Each Sector',fontsize=25)
prob_sector = prob_df['Likelihood_of_Good_Investment'].plot(kind='bar')

for each in prob_sector.patches:
    prob_sector.annotate(format(each.get_height(), '.2f'),
                   (each.get_x() + each.get_width() / 2,
                   each.get_height()),ha='center',va='center',
                   size=20, xytext=(0,-10),
                   textcoords='offset points')
plt.xticks(fontsize=14,rotation=65)
plt.show()
```



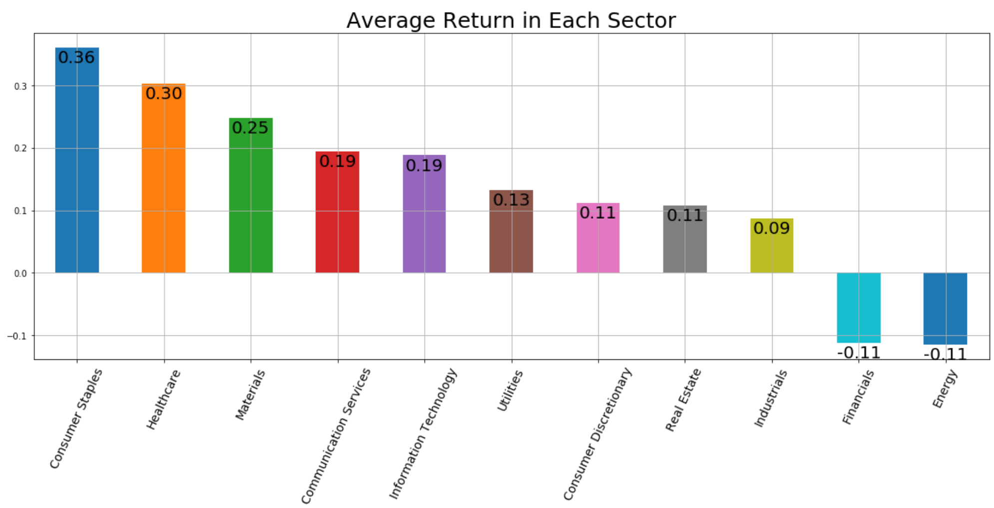
In [25]:

```
#Create visual for average return in each sector
plt.figure(figsize=(20,7))
plt.title('Average Return in Each Sector',fontsize=25)

sorted_avg_return_by_sector = real_deal_df.groupby('Sector').describe()['1-Year
sorted_return_plot = sorted_avg_return_by_sector.plot(kind='bar')

for each in sorted_return_plot.patches:
    sorted_return_plot.annotate(format(each.get_height(), '.2f'),
                    (each.get_x() + each.get_width() / 2,
                    each.get_height()),ha='center',va='center',
                    size=20, xytext=(0,-12),
                    textcoords='offset points')

plt.xticks(fontsize=14,rotation=65)
plt.xlabel(s='',fontsize=17)
plt.grid()
plt.show()
```

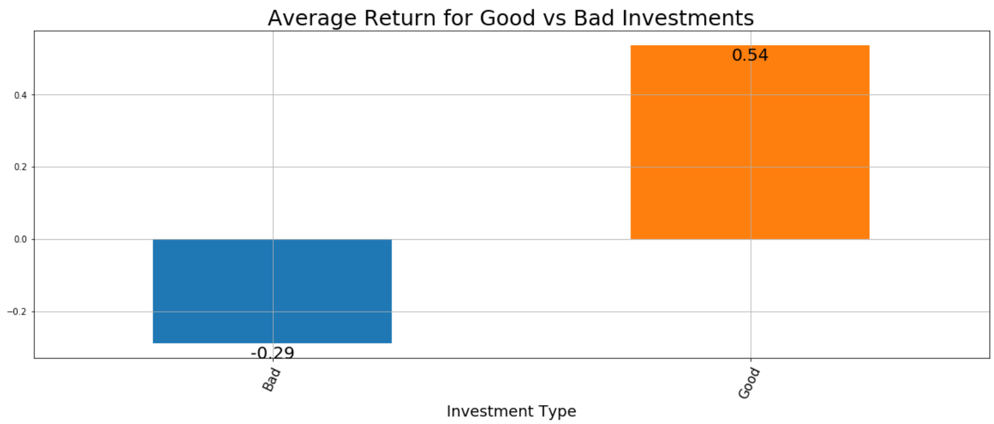


In [42]:

```
#Visualize average return for good vs bad investments
avg_return_good_and_bad = real_deal_df.groupby('Decision').describe()['1-Year_R
plt.figure(figsize=(20,7))
plt.title('Average Return for Good vs Bad Investments',fontsize=25)
avg_return_good_and_bad_plot = avg_return_good_and_bad.plot(kind='bar')

for each in avg_return_good_and_bad_plot.patches:
    avg_return_good_and_bad_plot.annotate(format(each.get_height(), '.2f'),
                   (each.get_x() + each.get_width() / 2,
                   each.get_height()),ha='center',va='center',
                   size=20, xytext=(0,-12),
                   textcoords='offset points')

plt.xlabel(s='Investment Type',fontsize=18)
plt.xticks(fontsize=15,rotation=65)
plt.grid()
plt.show()
```



In [180]:

```
#Stocks priced at 100 or less
real_deal_df[real_deal_df['P0']<=100].count()['Spinoff_Name']
```

Out[180]:

```
167
```

In [183]:

```
#Total stocks in final data set
real_deal_df['Spinoff_Name'].count()
```

Out[183]:

```
176
```

In [185]:

```
#Stocks priced at 100 or less and had a return of at least 10%
real_deal_df[(real_deal_df['P0']<=100) & (real_deal_df['1-Year_Return_As_Decima
```

Out[185]:

```
81
```

In [187]:

```
#Average return for stocks that are affordable and good investments
real_deal_df[(real_deal_df['P0']<=100) & (real_deal_df['1-Year_Return_As_Decima
#Average return for stocks that are affordable and good investments is 55%!!! T
```

Out[187]:

```
0.5461109409506173
```

In [186]:

```
#The real cream of the crop make up 46% of the data set. That's a pretty good s
#that is traditionally very niche and can be tricky to understand
real_deal_df[(real_deal_df['P0']<=100) & (real_deal_df['1-Year_Return_As_Decima
```

Out[186]:

```
0.4602272727272727
```

In [ ]:

```
#Metrics and Visuals for Data Analysis
#Split of investments into price point buckets -- check
#Total good investments vs bad -- check
#Probability of good investment on average per sector -- check
#Average return per sector -- check
#Average return for good vs bad investments -- check
#How many initial prices were under $100 USD? -- check
```