Machine Learning, a struktura kapitału spółek.

PROJEKT REKRUTACYJNY

ANNA MAZIARCZYK

O mnie

Anna Maziarczyk

Przez ostatnie 3 lata prowadziłam badania naukowe o kondycji finansowej Polskich spółek. Opierając się na wiedzy statystycznej publikowałam swoje wyniki w ogólnie dostępnych czasopismach. Posiadam wykształcenie matematyczne i finansowe, a od ostatniego roku interesuję się branżą IT i w tym kierunku sukcesywnie się uczę analizy danych w Python. Zdania, z którym się utożsamiam to „Jeśli czegoś nie wiem to się dowiem” i „wiedza nie jest niczym złym” dlatego lubię szukać odpowiedzi na pytania :)

Słowem wstępu

Jako, że w ostatnim czasie obracałam się w danych finansowych podejmuje wyzwanie zbudowania modelu, który będzie przewidywał strukturę kapitałową. Każda spółka musi zadbać o odpowiedni stosunek kapitału własnego i kapitału obcego aby czerpać jak najwięcej zysku. Jak wiemy jest to główny cel każdej działalności :)

Problem? Ale jaki problem?

Poznając różne modele Machine Learning wiemy, że definiując problem do rozwiązana a raczej pytanie na które chcemy poznać odpowiedź musimy określić czy jest to problem klasyfikacji czy regresji. Mój problem jest problemem regresji, która „umożliwia przewidywanie nieznanych wartości na podstawie znanych innych wartości". Chcę określić jak będzie kształtowała się *struktura kapitału*, która ma postać ilościową.

Ale może najpierw...

DANE

Dane pochodzą z bazy danych Notoria, w której dostępne są m.in. sprawozdania finansowe spółek notowanych na Giełdzie Papierów Wartościowych w Warszawie. Okres badania dotyczył 10 lat czyli okresu 2010-2019. Na podstawie potrzebnych danych tworzę własną bazę danych przy pomocy języka SQL i Management Studio po czym wszystko wrzucam do Google Colabolatory

	Source.Name.1	sektor	mediana	średnia zadł ogół	podatek	średnia ROA	ln SIZE	wsk rzecz ak trw/ao	wskaźnik dywidendy	ryzyko	średnia wsk dźwig fin	Struktura
0	AGORA	usługi	0.54	0.268842	0.629528	0.016473	2.660557	0.808878	0.033130	0.482471	1.366505	0.018795
1	AMICA	przemysł	0.46	0.591172	0.069862	0.047287	2.625140	0.486058	0.038090	0.198194	2.669319	0.042818
2	AMREST	usługi	0.54	0.585974	0.125818	0.049117	2.658260	0.864786	0.001041	0.153558	2.451094	0.041059
3	APATOR	przemysł	0.46	0.407365	0.154644	0.129215	2.561322	0.565156	0.102120	0.035926	1.706892	0.031450
4	APLISENS	przemysł	0.46	0.091589	0.151960	0.140450	2.435260	1.020143	0.054106	0.001844	1.101638	0.008021

Z bazy danych Notoria pobrałam interesujące mnie dane. Należą do nich:

Zmienna zależna:

- Struktura - relacja długu do majątku ogółem

Zmienne niezależne (predyktory):

- sektor - analizuję sektor przemysłu i usług
- mediana sektora - mediana zadłużenia sektora
- średnia zadłużenia ogółem -zobowiązania ogółem / aktywa ogółem
- podatek - podatek dochodowy /EBIT
- średnia ROA - rentowność aktywów
- ln SIZE - logarytm naturalny aktywów ogólem
- wsk rzecz ak trw/ao - wskaźnik rzeczowe aktywa trwałe / aktywa ogółem
- wskaźnik dywidendy - wypłacone dywidendy / aktywa ogółem
- ryzyko - odsetki wypłacone /EBIT
- dźwignia finansowa-wskaźnik dźwigni finansowej

Po kolei...

Po pobraniu danych do Google Colabolatory wszystkie operacje zostały wykonane przy pomocy języka python. Na początku pobieram wszystkie potrzebne biblioteki do obróbki tj. pandas, numy i tym razem seaborn do wizualizacji.

Po wstępnej obróbce okazuje się, że na zbiorze nie ma wielu obserwacji odstających czyli tzw. "outliersów". Wynika to z tego, że zbiór został już wcześniej przeze mnie przygotowany i opieram się tutaj już na danych bez braków danych.

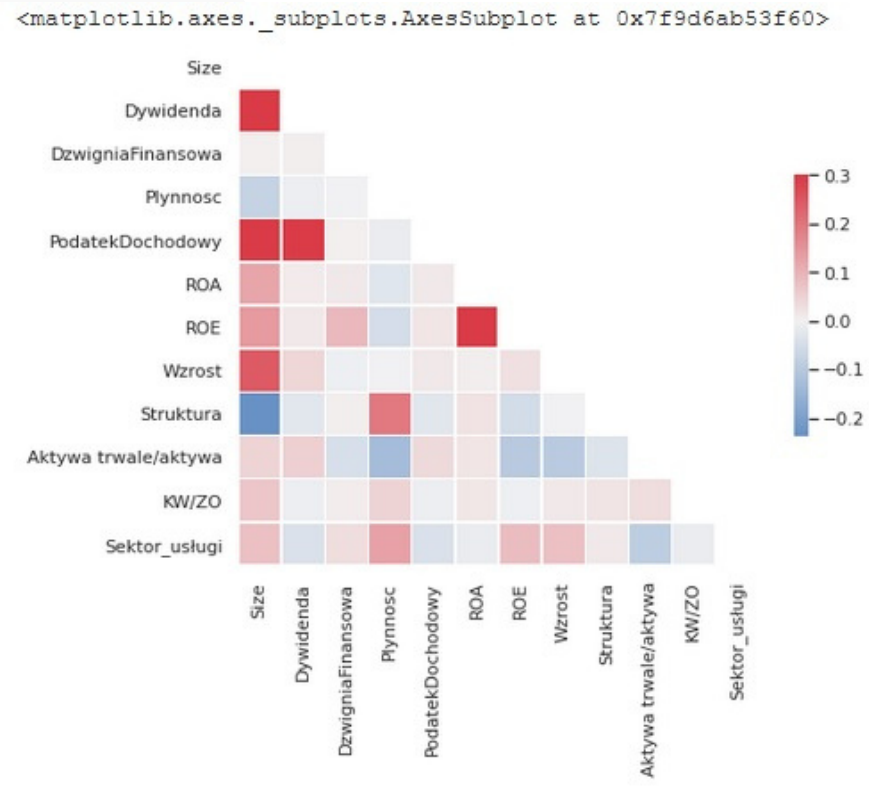
	count	mean	std	min	25%	50%	75%	max
mediana	147.0	0.486122	0.037644	0.460000	0.460000	0.460000	0.540000	0.540000
średnia zadł ogół	147.0	0.490188	0.189885	0.091589	0.359132	0.474585	0.592034	1.513936
podatek	147.0	0.273687	0.641086	0.001157	0.117073	0.169385	0.222351	6.942928
średnia ROA	147.0	0.032426	0.058864	-0.237710	0.005507	0.039743	0.061134	0.167009
ln SIZE	147.0	2.563755	0.125509	2.303283	2.468733	2.561136	2.651538	3.000861
wsk rzecz ak trw/ao	147.0	0.604101	0.452563	0.001196	0.230715	0.565156	0.945867	1.778996
wskaźnik dywidendy	147.0	0.056354	0.066539	0.000000	0.018659	0.045770	0.064634	0.548264
ryzyko	147.0	0.367895	0.775370	0.001844	0.071887	0.157211	0.343918	7.781722
średnia wsk dźwig fin	147.0	2.242022	1.400967	-5.590947	1.561740	1.891064	2.551311	9.203893
Struktura	147.0	0.037570	0.015131	0.008021	0.027885	0.035503	0.046035	0.134787

Rozpoczynając już docelową analizę mamy tutaj zestawienie podstawowych statystyk opisowych. Mój zbiór składa się zaledwie z 147 próbek co może wydawać się niewielkim zbiorem. Mimo wszystko podejmuję próbę i kontynuuje badanie dalej.

To dalej związek

W następnym kroku proponuję sprawdzić korelację między zmiennymi. Wykorzystuje w tym celu kolejną bibliotekę do wizualizacji czyli seaborn. Poszczególne wyniki przedstawiam już w postaci tzw. heatmapy czyli mapy ciepła. Na wykresie nie widać związku między zmiennymi. A jeżeli się pojawia to sił związku jest na niskim poziomie. W związku z tym wszystkie zmienne zostają w modelu.

```
sns.set(style="white")
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(8, 6))
cmap = sns.diverging_palette(250, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```



Jeszcze kodowanie

Ważnym krokiem przed podzieleniem zbioru na dane treningowe i testowe jest zakodowanie zmiennej kategorycznej. Zatem nie zapominając o tej istotnej zasadzie koduje zmienną "Sektor", która przyjmowała wartości "przemysł" lub " usługi" na wartości 0 lub 1. Kodowania dokonałam przy pomocy biblioteki pandas, a dokładnie pd.get_dummies() z ustawionym parametrem drop_first="True". Dzięki temu parametrowi otrzymujemy zmienne zakodowane w jednej kolumnie tj kolumnie usługi i 1 oznacza "tak", a 0 oznacza "nie".

Teraz trenujmy

Wykorzystuje train_test_split z modułu model_selection biblioteki Scikit-learn. Dodatkowo sprawdzam jak podzielił się zbiór danych.

```
# dziele dane na treningowe i testowe

from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test= train_test_split(data, target, random_state=42)

print(f'X_train shape {X_train.shape}')
print(f'y_train shape {y_train.shape}')
print(f'X_test shape {X_test.shape}')
print(f'y_test shape {y_test.shape}')
print(f'\nTest ratio: {len(X_test) / len(data):.2f}')
print(f'\ny_train:\n{y_train.value_counts()}')
print(f'\ny_test:\n{y_test.value_counts()}')
```

```
X_train shape (110, 10)
y_train shape (110,)
X_test shape (37, 10)
y_test shape (37,)

Test ratio: 0.25
```

Regresja, a może predykcja?

Wykorzystuję model regresji, który "umożliwia przewidywanie nieznanych wartości na podstawie innych znanych wartości". Dalej pozostaję przy bibliotece sklearn i z sklearn.linear_model importuję LinearRegression. Okazało się, że model jest zaskakująco dopasowany do danych. Predykcja modelu na poziomie 98%. I tutaj muszę zdradzić sekret... Nie bez powodu karzystam z tych zmiennych. Nauka zagraniczna już wie, że to się uda. Jednakże dla warunków polskich mogło być różnie :)

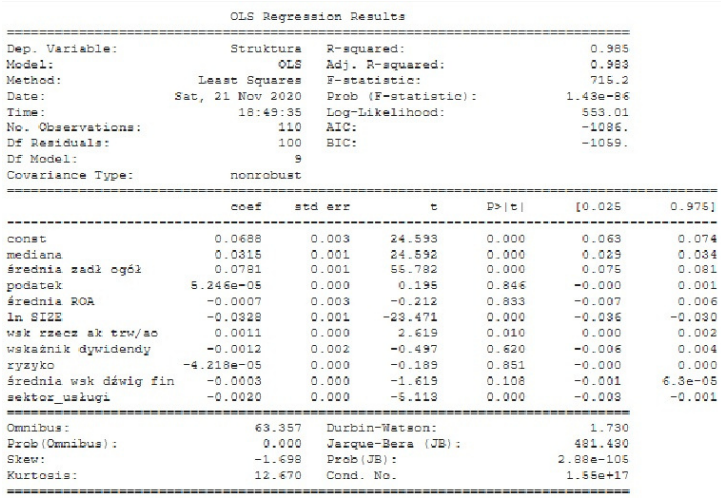
```
[ ] from sklearn.linear_model import LinearRegression
```

```
[ ] regressor = LinearRegression()
    regressor.fit(X_train, y_train)

    print(f'R2 score: {regressor.score(X_test, y_test):.4f}')
```

```
R2 score: 0.9717
```

Wynik jest zaskakująco wysoki jednak chciałam i sprawdziłam jeszcze eliminację wstaczną



OLS Regression Results

Dep. Variable:	Struktura	R-squared:	0.985
Model:	OLS	Adj. R-squared:	0.983
Method:	Least Squares	F-statistic:	715.2
Date:	Sat, 21 Nov 2020	Prob (F-statistic):	1.43e-86
Time:	18:49:35	Log-Likelihood:	553.01
No. Observations:	110	AIC:	-1086.
Df Residuals:	100	BIC:	-1059.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0688	0.003	24.593	0.000	0.063	0.074
mediana	0.0315	0.001	24.592	0.000	0.029	0.034
średnia zadł ogół	0.0781	0.001	55.782	0.000	0.075	0.081
podatek	5.246e-05	0.000	0.195	0.846	-0.000	0.001
średnia ROA	-0.0007	0.003	-0.212	0.833	-0.007	0.006
ln SIZE	-0.0328	0.001	-28.471	0.000	-0.035	-0.030
wsk rzecz ak trw/ac	0.0011	0.000	2.619	0.010	0.000	0.002
wskaźnik dywidendy	-0.0012	0.002	-0.497	0.620	-0.006	0.004
ryzyko	-4.218e-05	0.000	-0.189	0.851	-0.000	0.000
średnia wsk dźwig fin	-0.0003	0.000	-1.619	0.108	-0.001	6.3e-05
sektor_usługi	-0.0020	0.000	-5.113	0.000	-0.003	-0.001

Omnibus:	63.357	Durbin-Watson:	1.730
Prob(Omnibus):	0.000	Jarque-Bera (JB):	481.430
Skew:	-1.698	Prob(JB):	2.88e-105
Kurtosis:	12.670	Cond. No.	1.85e+17

Zaczynając od pozycji wyjściowej wykorzystałam bibliotekę statsmodels do kolejnych obliczeń statystycznych. Okazuje się, że nie wszystkie zmienne są istotne statystycznie z p<0,05 dlatego też w kolejnych krokach eliminuje poszczególne zmienne dla których p>0,05.

Po kilku takich krokach otrzymuje formułę koncową. Ponadto w każdym kroku wskaźnik Durbina-Watsona mieści się w przedziale optymalnym (tj.0-4). Udało się poprawić dopasowanie do 98,5%

Generuję model:

Y = 0,688 + 0,315* Mediana sektora + 0,0781* Zadłużenie ogółem - 0,033 * SIZE + 0,0011 * rzeczowe aktywa trwałe/aktywa ogółem

Pozostałe zmienne okazały się nieistotne dla modelu z p>0,05

Podsumowując

Metody Machine Learning mogą być wykorzystane również w modelowaniu finansów przedsiębiorstw. Do tej pory wszystkie testy statystyczne, badanie związku i szukanie istotnych różnic między zmiennymi wykonywałam w Statistice. Teraz misja "Python" :)

Myślę, że wyszło Git ale jeszcze dużo przede mną w zabawie z modelowaniem, pythonem i innymi narzędziami IT. Mimo wszystko jeżeli obudziłam ciekawość to kontak jest na dole :)

https://github.com/amaziarczyk/Projekt_rekrutacyjny

anna.maziarczyk@o2.pl