Project

Submitted by: Kanupriya Parashar

Panel Data: Analysis of ETF Arbitrage Oppurtunity

Qualitative Dependent Variable Model: Analysis of the Cardiovascular Disease Presence

In [1]:
```
import yfinance as yf
import statistics
import numpy as np
import pandas as pd
from pandas import DataFrame as df
import scipy
from scipy import io as spio
from scipy import stats
import matplotlib.pyplot as plt
from scipy import misc
from scipy.stats import norm
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
import seaborn as sns
sns.set(style="whitegrid")
import math
from BorutaShap import BorutaShap
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
import random
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
import linearmodels as plm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
```

1. Panel Data Models

We form a Panel Data with Adjusted Closing prices of S&P 500(GSPC), Dow Jones Industrial (DJI) and NASDAQ (IXIC). We aim to run three comparative models and find the arbitrage oppurutnites to invest in Exchange Traded Funds (ETF) backed against the S&P 500 index, seeing the movement and direction of the Adjusted Close prices of NASDAQ and Dow Jones Industrial indices.

In [2]:
```
indices_tickers=["^GSPC","^DJI","^IXIC"]
start_date="2019-12-31"
end_date="2023-01-01"
index_data=yf.download(indices_tickers,start=start_date,end=end_dat
index_data
```

```
[**********************100%%**********************]  3 of 3 complet
ed
```

Out[2]:

Date	^DJI	^GSPC	^IXIC
2019-12-31	28538.439453	3230.780029	8972.599609
2020-01-02	28868.800781	3257.850098	9092.190430
2020-01-03	28634.880859	3234.850098	9020.769531
2020-01-06	28703.380859	3246.280029	9071.469727
2020-01-07	28583.679688	3237.179932	9068.580078
...	...	...	...
2022-12-23	33203.929688	3844.820068	10497.860352
2022-12-27	33241.558594	3829.250000	10353.230469
2022-12-28	32875.710938	3783.219971	10213.290039
2022-12-29	33220.800781	3849.280029	10478.089844
2022-12-30	33147.250000	3839.500000	10466.480469

757 rows × 3 columns

In [3]:
```
#index_data=index_data.stack().reset_index().rename(columns={"level
```

In [4]:
```
index_data.head(100)
```

Out[4]:

Date	^DJI	^GSPC	^IXIC
2019-12-31	28538.439453	3230.780029	8972.599609
2020-01-02	28868.800781	3257.850098	9092.190430
2020-01-03	28634.880859	3234.850098	9020.769531
2020-01-06	28703.380859	3246.280029	9071.469727
2020-01-07	28583.679688	3237.179932	9068.580078
...	...	...	...
2020-05-18	24597.369141	2953.909912	9234.830078
2020-05-19	24206.859375	2922.939941	9185.099609
2020-05-20	24575.900391	2971.610107	9375.780273
2020-05-21	24474.119141	2948.510010	9284.879883
2020-05-22	24465.160156	2955.449951	9324.589844

100 rows × 3 columns

In [5]:
```
index_data.columns=["DJI","GSPC","IXIC"]
```

In [6]:
```
index_data.head()
```

Out[6]:

Date	DJI	GSPC	IXIC
2019-12-31	28538.439453	3230.780029	8972.599609
2020-01-02	28868.800781	3257.850098	9092.190430
2020-01-03	28634.880859	3234.850098	9020.769531
2020-01-06	28703.380859	3246.280029	9071.469727
2020-01-07	28583.679688	3237.179932	9068.580078

In [7]:
```
index_data.shape
```

Out[7]: (757, 3)

In [8]:
```
index_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 757 entries, 2019-12-31 to 2022-12-30
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   DJI     757 non-null    float64
 1   GSPC    757 non-null    float64
 2   IXIC    757 non-null    float64
dtypes: float64(3)
memory usage: 23.7 KB
```

In [9]:
```
index_data.isnull().any()
```

Out[9]:
```
DJI     False
GSPC    False
IXIC    False
dtype: bool
```

In [10]:
```
index_data.describe()
```

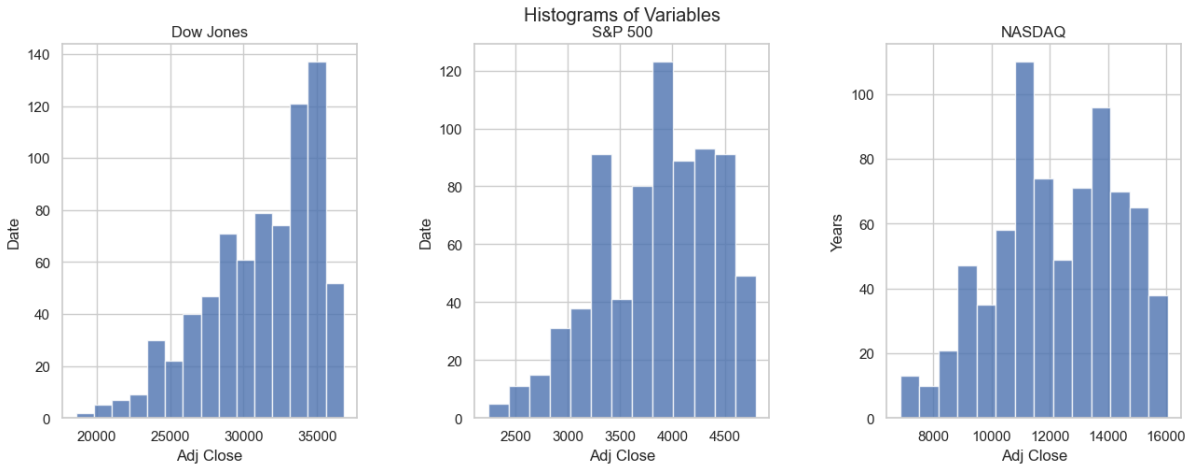
Out[10]:

	DJI	GSPC	IXIC
count	757.000000	757.000000	757.000000
mean	31269.544917	3861.253225	12261.137562
std	3724.274585	551.299490	2117.388188
min	18591.929688	2237.399902	6860.669922
25%	28745.089844	3401.199951	10829.500000
50%	32098.990234	3915.590088	12334.639648
75%	34364.500000	4319.939941	13972.530273
max	36799.648438	4796.560059	16057.440430

The table presents statistical measures for three financial indices: DJI, GSPC, and IXIC. With 757 data points, it outlines their central tendencies and dispersion. DJI's average stands at approximately 31,269, while GSPC and IXIC show means around 3,861 and 12,261, respectively. These figures, alongside minimum, maximum, and quartile values, offer insights into the distribution and variability of these indices over the specified period.

In [34]:
```
fig, ax = plt.subplots(1, 3, figsize = (15,5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac
fig.suptitle('Histograms of Variables')

ax[0].hist(index_data["DJI"], alpha = 0.8, bins = "fd")
ax[0].set_title("Dow Jones")
ax[0].set_xlabel("Adj Close")
ax[0].set_ylabel("Date")

ax[1].hist(index_data["GSPC"], alpha = 0.8, bins = "fd")
ax[1].set_title("S&P 500")
ax[1].set_xlabel("Adj Close")
ax[1].set_ylabel("Date")

ax[2].hist(index_data["IXIC"], alpha = 0.8, bins = "fd")
ax[2].set_title("NASDAQ")
ax[2].set_xlabel("Adj Close")
ax[2].set_ylabel("Years")
```

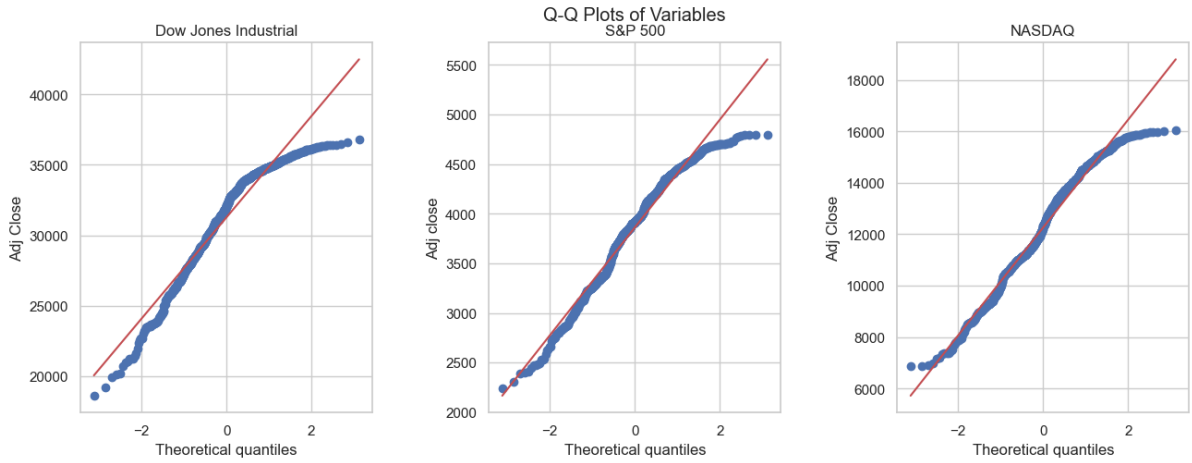
Out[34]:
```
Text(0, 0.5, 'Years')
```



We see that the data distribution is right skewed for all the indices. For NASDAQ the right skewness is less as compared to the other indices.

In [12]:
```
fig, ax = plt.subplots(1, 3, figsize = (15, 5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac
fig.suptitle('Q-Q Plots of Variables')

stats.probplot(index_data["DJI"], dist = "norm", plot = ax[0])
ax[0].set_title("Dow Jones Industrial")
ax[0].set_ylabel("Adj Close")

stats.probplot(index_data["GSPC"], dist = "norm", plot = ax[1])
ax[1].set_title("S&P 500")
ax[1].set_ylabel("Adj close")

stats.probplot(index_data["IXIC"], dist = "norm", plot = ax[2])
ax[2].set_title("NASDAQ")
ax[2].set_ylabel("Adj Close")
```

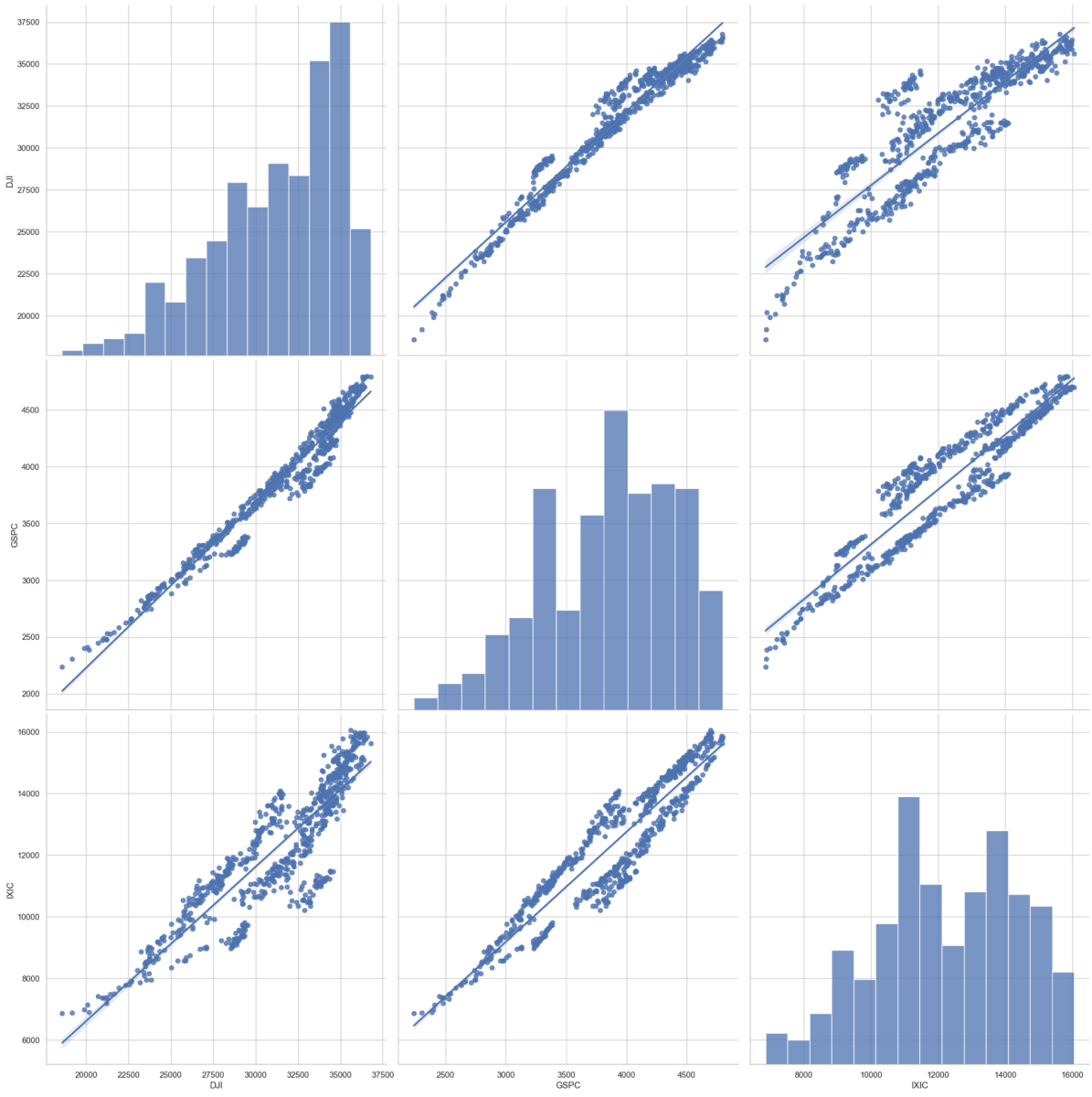
Out[12]:
```
Text(0, 0.5, 'Adj Close')
```



The QQ plot above show that there exists skewness in all the indices distrbution. Also none of the indices are normally distrbuted. The left tails in all the indices are deviating from normal distribution, also indicating presence of outliers in the dataset. For DJI even the right tail is deviating indicating the presence of outliers even in that direction.

In [13]:
```
plt.figure(figsize=(10,10))
sns.pairplot(index_data, height = 7.0, kind='reg')
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x16118f460>

<Figure size 1000x1000 with 0 Axes>
```



There seems to be a linear relationship between all pairs of the indices. We further give a strong support for this assessment below in our correlation matrix.

In [14]:
```
corr_matrix=index_data.corr()
corr_matrix
```

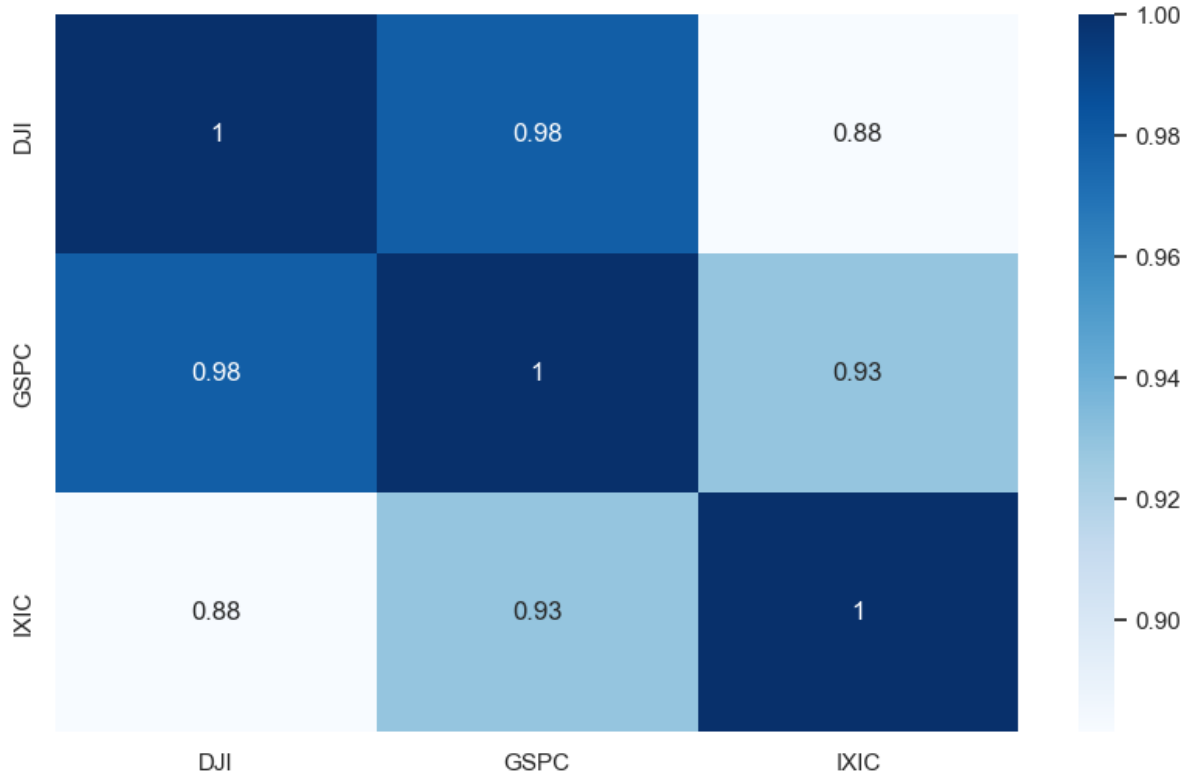
Out[14]:

	DJI	GSPC	IXIC
DJI	1.000000	0.979125	0.881534
GSPC	0.979125	1.000000	0.928735
IXIC	0.881534	0.928735	1.000000

Type *Markdown* and LaTeX: α^2

In [15]:
```
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix,annot=True, cmap="Blues")
```

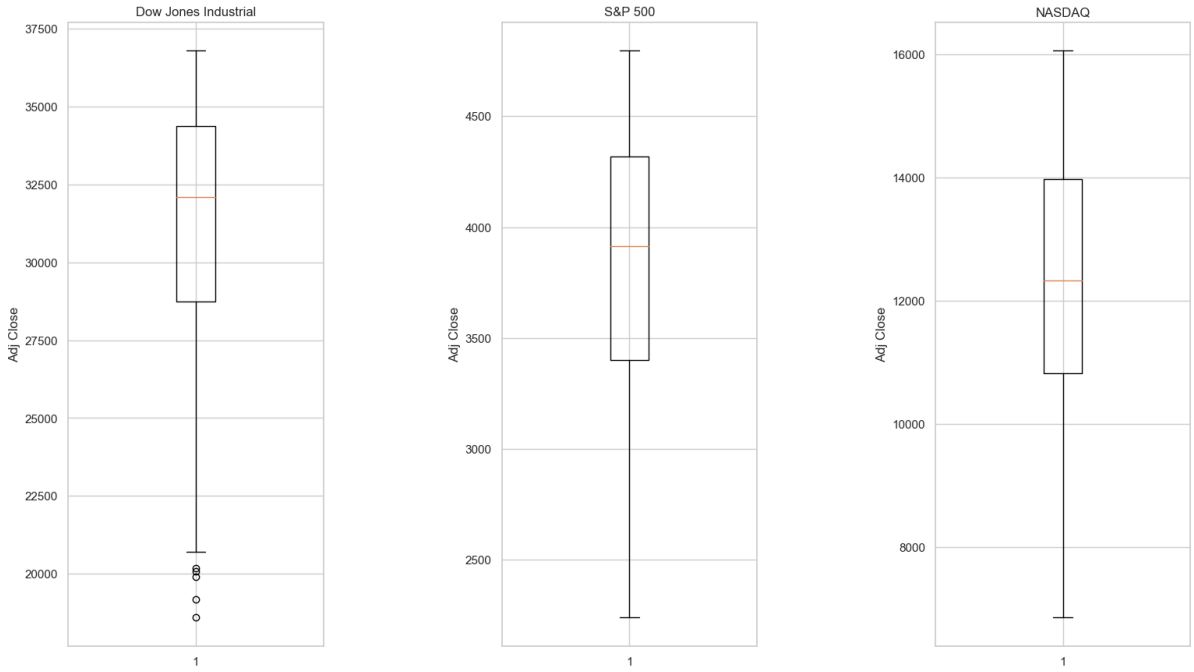
Out[15]: <AxesSubplot:>



There doesn't seem to be a perfect collinearity between the variables. Therefore, solving the multicollinearity problem in our model. but there exists strong positive collinearity between the variables. But that wouldn't be a problem in our multiple linear regression estimation, since, it doesn't violate the assumptions.

In [16]:
```
fig, ax = plt.subplots(1, 3, figsize = (18, 10))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac

ax[0].boxplot(index_data["DJI"])
ax[0].set_title("Dow Jones Industrial")
ax[0].set_ylabel("Adj Close")

ax[1].boxplot(index_data["GSPC"])
ax[1].set_title("S&P 500")
ax[1].set_ylabel("Adj Close")

ax[2].boxplot(index_data["IXIC"])
ax[2].set_title("NASDAQ")
ax[2].set_ylabel("Adj Close")
```

Out[16]:
```
Text(0, 0.5, 'Adj Close')
```



There is an overlap amongst the distribution for S&P 500, Dow Jones Industrial and NASDAQ. There is a right skewness in the distribution of all the variables. There are some potential outliers for Dow Jones Industrial. The spread for all three variables are almost the same. There is not much significant difference between the distribution of the three variables. The median value for Dow Jones Industrial is higher than for S&P 500 and NASDAQ.

Pooled Model

In [17]:
```
index_data.reset_index(inplace=True)
index_data['year'] = index_data['Date'].dt.year


index_data['nr'] = range(len(index_data))

index_data = index_data.set_index(['nr', 'year'], drop=False)
index_data
```

Out[17]:

		Date	DJI	GSPC	IXIC	year	nr
nr	**year**						
0	**2019**	2019-12-31	28538.439453	3230.780029	8972.599609	2019	0
1	**2020**	2020-01-02	28868.800781	3257.850098	9092.190430	2020	1
2	**2020**	2020-01-03	28634.880859	3234.850098	9020.769531	2020	2
3	**2020**	2020-01-06	28703.380859	3246.280029	9071.469727	2020	3
4	**2020**	2020-01-07	28583.679688	3237.179932	9068.580078	2020	4
...	**...**	...	...	...	...	...	...
752	**2022**	2022-12-23	33203.929688	3844.820068	10497.860352	2022	752
753	**2022**	2022-12-27	33241.558594	3829.250000	10353.230469	2022	753
754	**2022**	2022-12-28	32875.710938	3783.219971	10213.290039	2022	754
755	**2022**	2022-12-29	33220.800781	3849.280029	10478.089844	2022	755
756	**2022**	2022-12-30	33147.250000	3839.500000	10466.480469	2022	756

757 rows × 6 columns

In [18]:
```
index_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 757 entries, (0, 2019) to (756, 2022)
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    757 non-null    datetime64[ns]
 1   DJI     757 non-null    float64       
 2   GSPC    757 non-null    float64       
 3   IXIC    757 non-null    float64       
 4   year    757 non-null    int32         
 5   nr      757 non-null    int64         
dtypes: datetime64[ns](1), float64(3), int32(1), int64(1)
memory usage: 57.1 KB
```

In [19]:
```
index_data.isnull().any()
```

Out[19]:
```
Date    False
DJI     False
GSPC    False
IXIC    False
year    False
nr      False
dtype: bool
```

In [25]:
```
model1 = plm.PooledOLS.from_formula(
    formula='GSPC ~ DJI + IXIC', data=index_data)
results1 = model1.fit()
results1
```

Out[25]:

PooledOLS Estimation Summary

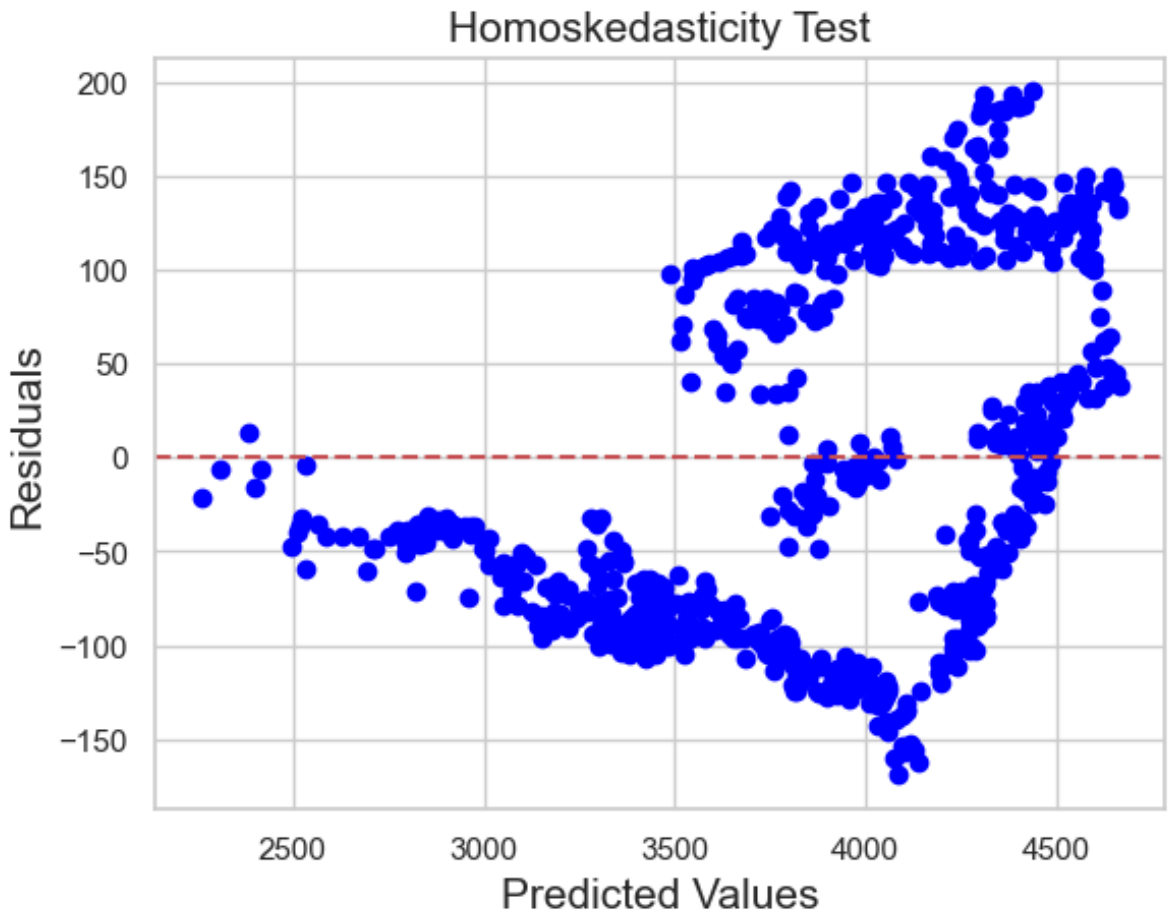
Dep. Variable:	GSPC	**R-squared:**	0.9994
Estimator:	PooledOLS	**R-squared (Between):**	0.9994
No. Observations:	757	**R-squared (Within):**	0.0000
Date:	Wed, Dec 06 2023	**R-squared (Overall):**	0.9994
Time:	14:07:42	**Log-likelihood**	-4507.3
Cov. Estimator:	Unadjusted		
		F-statistic:	6.601e+05
Entities:	757	**P-value**	0.0000
Avg Obs:	1.0000	**Distribution:**	F(2,755)
Min Obs:	1.0000		
Max Obs:	1.0000	**F-statistic (robust):**	6.601e+05
		P-value	0.0000
Time periods:	4	**Distribution:**	F(2,755)
Avg Obs:	189.25		
Min Obs:	1.0000		
Max Obs:	253.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
DJI	0.0870	0.0012	69.898	0.0000	0.0846	0.0894
IXIC	0.0934	0.0032	29.664	0.0000	0.0873	0.0996

id: 0x17348ada0

In [27]:
```
#checking heteroscedasticity
fig, ax = plt.subplots()
ax.scatter(results1.fitted_values, results1.resids, color = 'blue')
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicted Values', fontsize = 15)
ax.set_ylabel('Residuals', fontsize = 15)
ax.set_title('Homoskedasticity Test', fontsize = 15)
plt.show()
```



The graph above plotting Predicted Values against the Residuals clearly show heteroscedasticity, but we would do further investigation to confirm the same.

In [32]:
```
from statsmodels.stats.diagnostic import het_white, het_breuschpaga
import statsmodels.api as sm


# Extract residuals and exogenous variables
residuals1 = results1.resids
exog_vars = index_data[['DJI', 'IXIC']]
exog = sm.add_constant(exog_vars)


# Breusch-Pagan test for heteroscedasticity
breusch_pagan_test_results = het_breuschpagan(residuals1, exog)
bp_F_stat, bp_p_value = breusch_pagan_test_results[2], breusch_paga
print(f"\nBreusch-Pagan Test F-stat: {bp_F_stat}, P-value: {bp_p_va
```

```
Breusch-Pagan Test F-stat: 31.51934081364031, P-value: 7.137240981
2457e-14
```

Since the p-value is small and the F-statistics is also small, it indicates that we reject the null hypothesis and confirm that there is heteroscedasticity in our Pooled panel model.

In [35]:
```
#Durbin Watson test for autocorrelation
from statsmodels.stats.stattools import durbin_watson

durbin_watson_test_results = durbin_watson(residuals1)
print(durbin_watson_test_results)
```

```
0.006537988032687623
```

A Durbin-Watson statistic of 0.006537988032687623, which is extremely close to 0, indicates strong positive autocorrelation in the residuals. Positive autocorrelation means that adjacent residuals are highly positively correlated, violating the assumption of independence between residuals.

Since the Pooled panel model violates its assumption for homoscedasticity and autocorrelation, we will not consider the model for violating its assumptions.

Fixed Effects Model

In [39]:
```
model2 = plm.PanelOLS.from_formula(
    formula='GSPC ~ DJI + IXIC', data=index_data, drop_absorbed=Tru
results2 = model2.fit()
results2
```

Out[39]:

PanelOLS Estimation Summary

Dep. Variable:	GSPC	**R-squared:**	0.9994
Estimator:	PanelOLS	**R-squared (Between):**	0.9994
No. Observations:	757	**R-squared (Within):**	0.0000
Date:	Wed, Dec 06 2023	**R-squared (Overall):**	0.9994
Time:	16:29:51	**Log-likelihood**	-4507.3
Cov. Estimator:	Unadjusted		
		F-statistic:	6.601e+05
Entities:	757	**P-value**	0.0000
Avg Obs:	1.0000	**Distribution:**	F(2,755)
Min Obs:	1.0000		
Max Obs:	1.0000	**F-statistic (robust):**	6.601e+05
		P-value	0.0000
Time periods:	4	**Distribution:**	F(2,755)
Avg Obs:	189.25		
Min Obs:	1.0000		
Max Obs:	253.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
DJI	0.0870	0.0012	69.898	0.0000	0.0846	0.0894
IXIC	0.0934	0.0032	29.664	0.0000	0.0873	0.0996

id: 0x28c4a38b0

Excluded the Entity Effects as it was abosorbing too much variance. Better estimation can be seen on removing it.

In [40]:
```
#with time entity
model7= plm.PanelOLS.from_formula(
    formula='GSPC ~ DJI + IXIC + TimeEffects', data=index_data).fit
model7
```

Out[40]:

PanelOLS Estimation Summary

Dep. Variable:	GSPC	**R-squared:**	0.9697
Estimator:	PanelOLS	**R-squared (Between):**	0.9990
No. Observations:	757	**R-squared (Within):**	0.0000
Date:	Wed, Dec 06 2023	**R-squared (Overall):**	0.9990
Time:	16:30:24	**Log-likelihood**	-4065.6
Cov. Estimator:	Unadjusted		
		F-statistic:	1.203e+04
Entities:	757	**P-value**	0.0000
Avg Obs:	1.0000	**Distribution:**	F(2,751)
Min Obs:	1.0000		
Max Obs:	1.0000	**F-statistic (robust):**	1.203e+04
		P-value	0.0000
Time periods:	4	**Distribution:**	F(2,751)
Avg Obs:	189.25		
Min Obs:	1.0000		
Max Obs:	253.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
DJI	0.0754	0.0016	48.249	0.0000	0.0724	0.0785
IXIC	0.1286	0.0025	51.577	0.0000	0.1237	0.1335

F-test for Poolability: 367.39
P-value: 0.0000
Distribution: F(3,751)

Included effects: Time
id: 0x28cc54cd0

We will also not consider the model with both Time and Entity Effects becasue it absorbs alot variance and doesn't give an estimation.

In [43]:
```
from linearmodels.panel import compare

comparison = compare([results2,model7])


print(comparison)
```

```
                  Model Comparison
=================================================
                             Model 0        Model 1
-------------------------------------------------
Dep. Variable                   GSPC           GSPC
Estimator                   PanelOLS       PanelOLS
No. Observations                 757            757
Cov. Est.                Unadjusted     Unadjusted
R-squared                    0.9994         0.9697
R-Squared (Within)           0.0000         0.0000
R-Squared (Between)          0.9994         0.9990
R-Squared (Overall)          0.9994         0.9990
F-statistic                6.601e+05      1.203e+04
P-value (F-stat)             0.0000         0.0000
====================     ============   ============
DJI                          0.0870         0.0754
                            (69.898)       (48.249)
IXIC                         0.0934         0.1286
                            (29.664)       (51.577)
======================== ============== ==============
Effects                                         Time
-------------------------------------------------

T-stats reported in parentheses
```

Based on this comparison of the two model, one where we don't take either entity or time efefcts and one where we only take time effects, we see that the R squared is better for the model2 (sans entity and time effects), where 99.94% of the GSPC Adj Close is explained by the Adj Close of IXIC and DJI. We also see that the F-statistics, which is significant, is better for the model2 than model7, indicating overall significance of the model2. The coefficients in both the models are positive but the coefficient for IXIC changes in the two models, but for model2, the Standard Errors associated with that model is less, giving better reasoning to consider model2.

Random Effects

In [22]:
```
#taking Adj Close prices for Random Effects
import linearmodels as plm

try:
    model4 = plm.RandomEffects.from_formula(formula='GSPC ~ DJI + I
    results4 = model4.fit()
    print(results3)
except ZeroDivisionError:
    print("Zero Division Error occurred during model estimation.")
```

```
Zero Division Error occurred during model estimation.
```

In [11]:
```
#taking returns and seeing if random effects works on that
indices_tickers=["^GSPC","^DJI","^IXIC"]
start_date="2019-12-31"
end_date="2023-01-01"
index_data1=yf.download(indices_tickers,start=start_date,end=end_da
index_data1
```

```
[**********************100%%**********************]  3 of 3 complet
ed
```

Out[11]:

Date	^DJI	^GSPC	^IXIC
2019-12-31	NaN	NaN	NaN
2020-01-02	0.011576	0.008379	0.013328
2020-01-03	-0.008103	-0.007060	-0.007855
2020-01-06	0.002392	0.003533	0.005620
2020-01-07	-0.004170	-0.002803	-0.000319
...	...	...	...
2022-12-23	0.005342	0.005868	0.002075
2022-12-27	0.001133	-0.004050	-0.013777
2022-12-28	-0.011006	-0.012021	-0.013517
2022-12-29	0.010497	0.017461	0.025927
2022-12-30	-0.002214	-0.002541	-0.001108

757 rows × 3 columns

In [12]:
```
index_data1.columns=["DJI","GSPC","IXIC"]
```

In [13]:
```
index_data1.dropna(inplace=True)
```

In [14]:
```
index_data1.columns
```

Out[14]:
```
Index(['DJI', 'GSPC', 'IXIC'], dtype='object')
```

In [15]:
```
index_data1.reset_index(inplace=True)
index_data1['year'] = index_data1['Date'].dt.year


index_data1['nr'] = range(len(index_data1))

index_data1 = index_data1.set_index(['nr', 'year'], drop=False)
index_data1
```

Out[15]:

		Date	DJI	GSPC	IXIC	year	nr
nr	**year**						
0	**2020**	2020-01-02	0.011576	0.008379	0.013328	2020	0
1	**2020**	2020-01-03	-0.008103	-0.007060	-0.007855	2020	1
2	**2020**	2020-01-06	0.002392	0.003533	0.005620	2020	2
3	**2020**	2020-01-07	-0.004170	-0.002803	-0.000319	2020	3
4	**2020**	2020-01-08	0.005647	0.004902	0.006689	2020	4
...	**...**	...	...	...	...	...	...
751	**2022**	2022-12-23	0.005342	0.005868	0.002075	2022	751
752	**2022**	2022-12-27	0.001133	-0.004050	-0.013777	2022	752
753	**2022**	2022-12-28	-0.011006	-0.012021	-0.013517	2022	753
754	**2022**	2022-12-29	0.010497	0.017461	0.025927	2022	754
755	**2022**	2022-12-30	-0.002214	-0.002541	-0.001108	2022	755

756 rows × 6 columns

In [16]:
```
import linearmodels as plm

try:
    model3 = plm.RandomEffects.from_formula(formula='GSPC ~ DJI + I
    results3 = model3.fit()
    print(results3)
except ZeroDivisionError:
    print("Zero Division Error occurred during model estimation.")
```

```
Zero Division Error occurred during model estimation.
```

We will drop the Ramndom Effects model when we take both Adjusted Close price and Returns, and we see the problem that there exists Zero Division Error. There seems to be division by zero since there is not much variance between the values of the indices. Since, we would not like to interfere with the intergity of the data from Yahoo Finance, we will retain the originality of the data, and conclude that the Random Effects Model doesn't work for estimating the relationship in our model. We will continue with our analysis with two models i.e., Pooled Model and Fixed Effects model.

Final Model

As our final model we will be considering the Fixed Effects model (model2) where we have taken neither time or entity effects. Based on the comparison above for the two types of Fixed Effects model, the model2 (sans the effects) give us the best estimation and goodness of fit. It shows that both DJI and IXIC adjusted close prices have a positive imapct on the GSPC adjusted close prices. The IXIC has more stregth in its influence on the GSPC adjusted close price. The coefficients of this model are also statiscally significant. This positive relationship between these indices, could suggest investment strategies leveraging the information to balance a portfolio or exploit market trends for potential arbitrage opportunities.

2. Qualitative Dependent Variable Models

Data taken from the UCI Machine Learning Repository. It is an assessment of presence of Cardiovascular Disease, here the term "cardio" indicated as a binary variable, indicated the presence of cardiovascular disease, where 1 indicates presence of cardiovascular disease and 0 indicates the absence of the same. In our analysis we wish to see which lifestyle choices and co-morbidities can increase probability of cardiovascular disease. In our models we will be taking, "cardio" as our dependent variable and "weight", to indicate obesity, "ap_hi" and "ap_lo" to indicate Systolic and Diastolic Blood Pressure respectively, "cholesterol", to indicate different level of cholesterol, "glucose", to indicate differenct levels of glucose, "smoke", "alco", to indicate smoking and alcohol consumption by people and "active" to show whether the individuals are following a active lifestyle.

In [22]:
```
dt=pd.read_csv("cardio_train.csv")
dt.head()
```

Out[22]:

	id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;cardio
0	0;18393;2;168;62.0;110;80;1;1;0;0;1;0
1	1;20228;1;156;85.0;140;90;3;1;0;0;1;1
2	2;18857;1;165;64.0;130;70;3;1;0;0;0;1
3	3;17623;2;169;82.0;150;100;1;1;0;0;1;1
4	4;17474;1;156;56.0;100;60;1;1;0;0;0;0

In [23]:
```
dt.columns
```

Out[23]:
```
Index(['id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;s
moke;alco;active;cardio'], dtype='object')
```

In [24]:
```
dt = pd.read_csv("cardio_train.csv", delimiter=';')


combined_column_string = 'id;age;gender;height;weight;ap_hi;ap_lo;c
individual_columns = combined_column_string.split(';')

dt.columns = individual_columns

print(dt.head())
```

```
   id    age  gender  height  weight  ap_hi  ap_lo  cholesterol  g
luc  smoke  \
0   0  18393       2     168    62.0    110     80            1
1       0
1   1  20228       1     156    85.0    140     90            3
1       0
2   2  18857       1     165    64.0    130     70            3
1       0
3   3  17623       2     169    82.0    150    100            1
1       0
4   4  17474       1     156    56.0    100     60            1
1       0

   alco  active  cardio
0     0       1       0
1     0       1       1
2     0       0       1
3     0       1       1
4     0       0       0
```

In [25]:
```
dt.head()
```

Out[25]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	0	18393	2	168	62.0	110	80	1	1	0	0	1
1	1	20228	1	156	85.0	140	90	3	1	0	0	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0
3	3	17623	2	169	82.0	150	100	1	1	0	0	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0

In [26]:
```
dt.shape
```

Out[26]:
```
(70000, 13)
```

In [27]:
```
dt.isnull().any()
```

Out[27]:
```
id             False
age            False
gender         False
height         False
weight         False
ap_hi          False
ap_lo          False
cholesterol    False
gluc           False
smoke          False
alco           False
active         False
cardio         False
dtype: bool
```

In [28]:
```
dt.dropna()
```

Out[28]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco
0	0	18393	2	168	62.0	110	80	1	1	0	0
1	1	20228	1	156	85.0	140	90	3	1	0	0
2	2	18857	1	165	64.0	130	70	3	1	0	0
3	3	17623	2	169	82.0	150	100	1	1	0	0
4	4	17474	1	156	56.0	100	60	1	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...
69995	99993	19240	2	168	76.0	120	80	1	1	1	0
69996	99995	22601	1	158	126.0	140	90	2	2	0	0
69997	99996	19066	2	183	105.0	180	90	3	1	0	1
69998	99998	22431	1	163	72.0	135	80	1	2	0	0
69999	99999	20540	1	170	72.0	120	80	2	1	0	0

70000 rows × 13 columns

In [58]:
```
dt=dt[["cardio","weight","ap_hi","ap_lo","cholesterol","gluc","smok
dt.head()
```

Out[58]:

	cardio	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	0	62.0	110	80	1	1	0	0	1
1	1	85.0	140	90	3	1	0	0	1
2	1	64.0	130	70	3	1	0	0	0
3	1	82.0	150	100	1	1	0	0	1
4	0	56.0	100	60	1	1	0	0	0

In [60]:
```
dt.shape
```

Out[60]:
```
(70000, 9)
```

In [35]:
```
dt.describe()
```

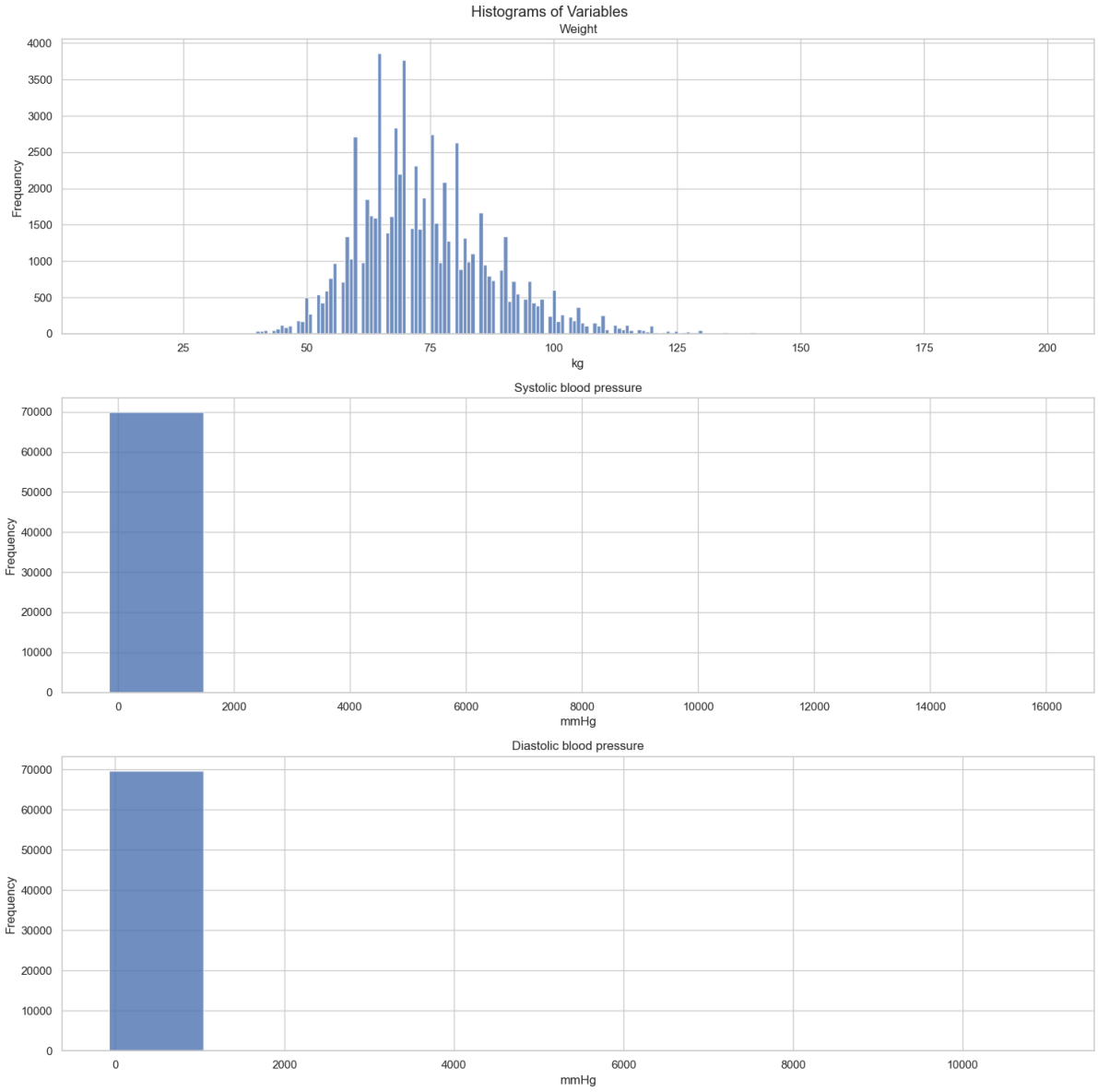
Out[35]:

	cardio	weight	ap_hi	ap_lo	cholesterol	gluc
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	0.499700	74.205690	128.817286	96.630414	1.366871	1.226457
std	0.500003	14.395757	154.011419	188.472530	0.680250	0.572270
min	0.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000
25%	0.000000	65.000000	120.000000	80.000000	1.000000	1.000000
50%	0.000000	72.000000	120.000000	80.000000	1.000000	1.000000
75%	1.000000	82.000000	140.000000	90.000000	2.000000	1.000000
max	1.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000

There are 70,000 observations in this dataset. The mean value of the 'cardio' variable is approximately 0.5, suggesting that on average, there's an almost equal split between the two categories (0 and 1). The average weight is around 74.2 kg, with a standard deviation of approximately 14.4 kg. The mean systolic blood pressure is approximately 128.8 mmHg, with a significantly high standard deviation (154.0 mmHg). The mean diastolic blood pressure is about 96.6 mmHg, with a high standard deviation of 188.5 mmHg. The mean for cholesterol and glucose represent that 136% and 122% respectively are observed in the data. It also shows that on average 8.8% people smoke, 5.3% people consume alcohol and 80.3% partake in activity as a lifestyle choice.

In [38]:
```
fig, ax = plt.subplots(3, 1, figsize=(15, 15))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac
fig.suptitle('Histograms of Variables')



# Weight
ax[0].hist(dt["weight"], alpha=0.8, bins="fd")
ax[0].set_title("Weight")
ax[0].set_xlabel("kg")
ax[0].set_ylabel("Frequency")

# Systolic Blood Pressure (ap_hi)
ax[1].hist(dt["ap_hi"], alpha=0.8, bins=10)
ax[1].set_title("Systolic blood pressure")
ax[1].set_xlabel("mmHg")
ax[1].set_ylabel("Frequency")

# Diastolic Blood Pressure (ap_lo)
ax[2].hist(dt["ap_lo"], alpha=0.8, bins=10)
ax[2].set_title("Diastolic blood pressure")
ax[2].set_xlabel("mmHg")
ax[2].set_ylabel("Frequency")
```

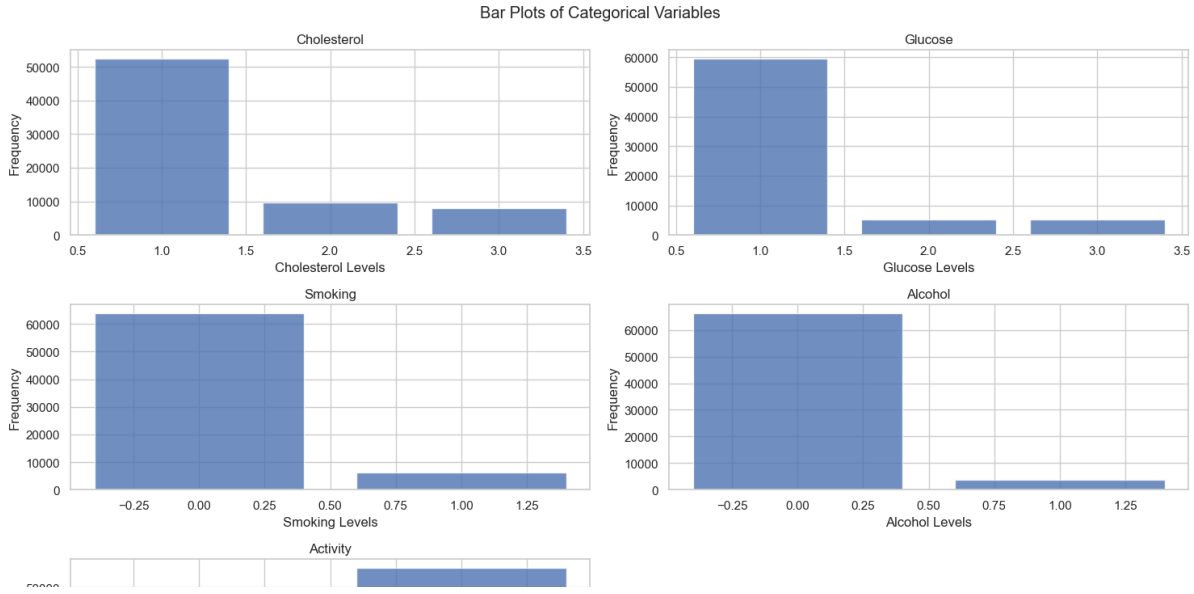
```
plt.tight_layout()
plt.show()
```

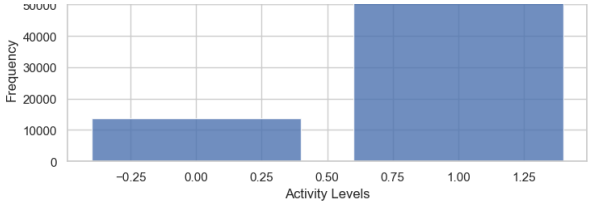


From the histograms above, we can observe that the data distribution for weight is almost symmetric with a slight left skewness, which will be verified later in the QQ plots. For the Systolic and Diastolic BP, since the values are restricted to the range of blood pressure metrics, they are rather a bar representing the values, concentrated in that region.

In [144]:
```
fig, ax = plt.subplots(3, 2, figsize=(15, 10))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac
fig.suptitle('Bar Plots of Categorical Variables')

# Cholesterol
cholesterol_counts = dt['cholesterol'].value_counts()
ax[0, 0].bar(cholesterol_counts.index, cholesterol_counts.values, a
```

```
ax[0, 0].set_title('Cholesterol')
ax[0, 0].set_xlabel('Cholesterol Levels')
ax[0, 0].set_ylabel('Frequency')

# Glucose
glucose_counts = dt['gluc'].value_counts()
ax[0, 1].bar(glucose_counts.index, glucose_counts.values, alpha=0.8
ax[0, 1].set_title('Glucose')
ax[0, 1].set_xlabel('Glucose Levels')
ax[0, 1].set_ylabel('Frequency')

# Smoke
smoke_counts = dt['smoke'].value_counts()
ax[1, 0].bar(smoke_counts.index, smoke_counts.values, alpha=0.8)
ax[1, 0].set_title('Smoking')
ax[1, 0].set_xlabel('Smoking Levels')
ax[1, 0].set_ylabel('Frequency')

# Alcohol
alcohol_counts = dt['alco'].value_counts()
ax[1, 1].bar(alcohol_counts.index, alcohol_counts.values, alpha=0.8
ax[1, 1].set_title('Alcohol')
ax[1, 1].set_xlabel('Alcohol Levels')
ax[1, 1].set_ylabel('Frequency')

# Active
active_counts = dt['active'].value_counts()
ax[2, 0].bar(active_counts.index, active_counts.values, alpha=0.8)
ax[2, 0].set_title('Activity')
ax[2, 0].set_xlabel('Activity Levels')
ax[2, 0].set_ylabel('Frequency')

ax[2, 1].axis('off')

plt.tight_layout()
plt.show()
```

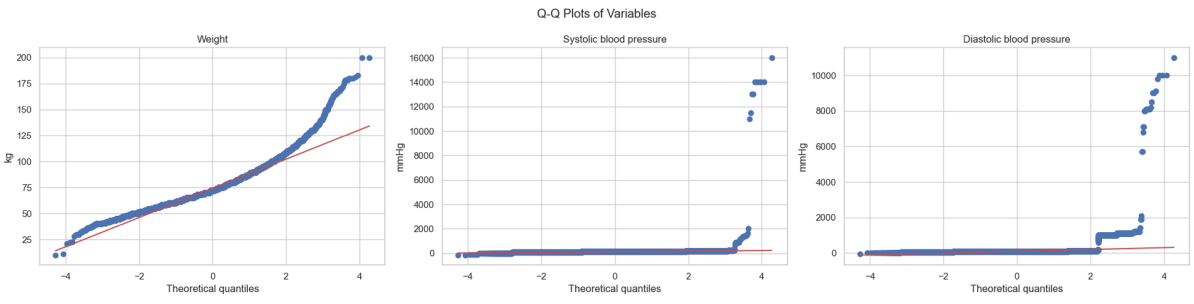




We can clearly observe that for cholesterol, there are more number of observations for normal level of cholesterol and almost equal frequency for those above normal and well above normal levels. The same observation can be made for the glucose levels in individuals, where there are more people with more normal levels of glucose. We also see that there are less people in our dataset who are smoking and consume alcohol. In the case of activity, we can see that there are more people who are active.

In [43]:

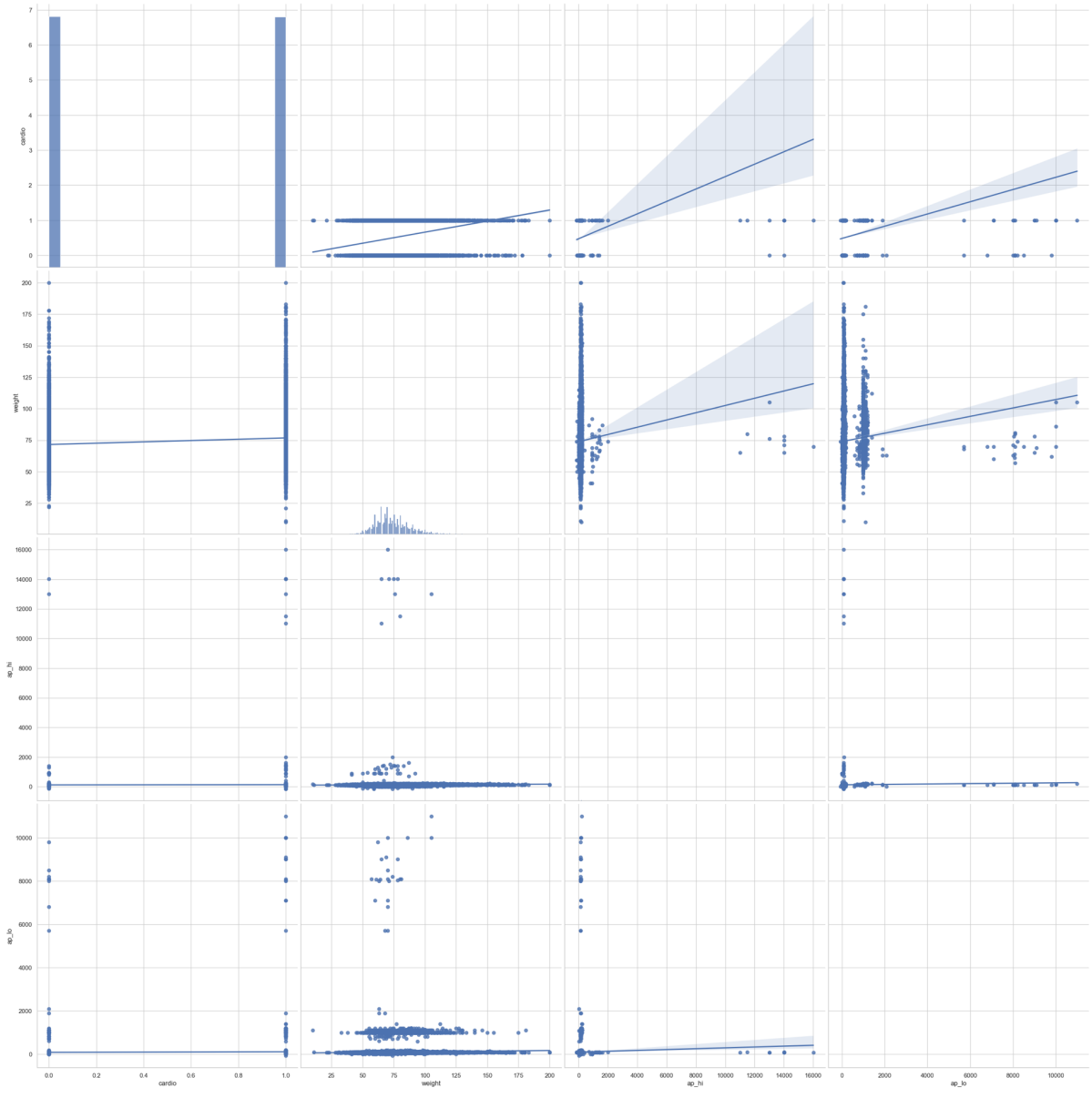
```
fig, ax = plt.subplots(1, 3, figsize=(20, 5))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac
fig.suptitle('Q-Q Plots of Variables')

stats.probplot(dt["weight"], dist="norm", plot=ax[0])
ax[0].set_title("Weight")
ax[0].set_ylabel("kg")

stats.probplot(dt["ap_hi"], dist="norm", plot=ax[1])
ax[1].set_title("Systolic blood pressure")
ax[1].set_ylabel("mmHg")

stats.probplot(dt["ap_lo"], dist="norm", plot=ax[2])
ax[2].set_title("Diastolic blood pressure")
ax[2].set_ylabel("mmHg")

plt.tight_layout()
plt.show()
```



We can clearly observe that the weight variable is not normally distributed, the left tail of the observations devaites indicating left skewness. On the other hand for Systolic and Diastolic BP, we can see, that it is also not normally distributed, there is again a left tail devaition, indicating left skewness even in the distribution fo these two variables.

In [136]:
```
plt.figure(figsize=(20,20))
sns.pairplot(dt, height = 7.0, kind='reg')
```

Out[136]:
```
<seaborn.axisgrid.PairGrid at 0x2b100f1c0>

<Figure size 2000x2000 with 0 Axes>
```



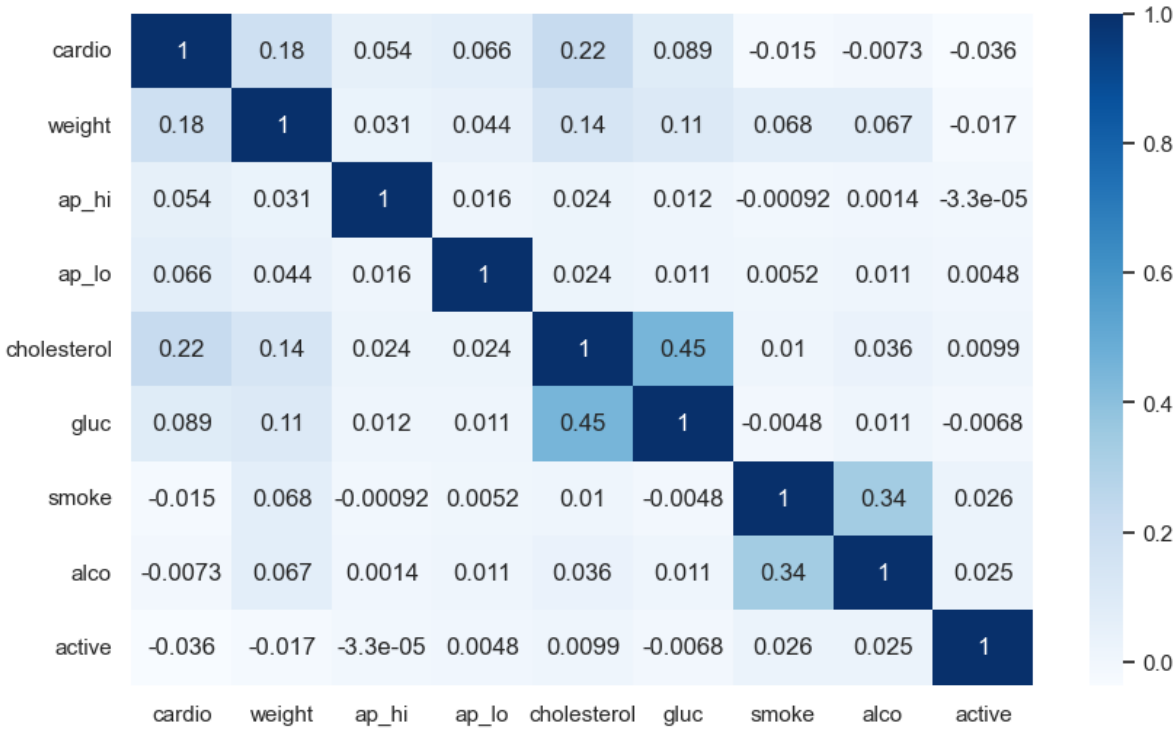
In [46]:
```
corr_matrix1=dt.corr()
corr_matrix1
```

Out[46]:

	cardio	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	
cardio	1.000000	0.181660	0.054475	0.065719	0.221147	0.089307	-0.015486	-0.
weight	0.181660	1.000000	0.030702	0.043710	0.141768	0.106857	0.067780	0.
ap_hi	0.054475	0.030702	1.000000	0.016086	0.023778	0.011841	-0.000922	0.
ap_lo	0.065719	0.043710	0.016086	1.000000	0.024019	0.010806	0.005186	0.
cholesterol	0.221147	0.141768	0.023778	0.024019	1.000000	0.451578	0.010354	0.
gluc	0.089307	0.106857	0.011841	0.010806	0.451578	1.000000	-0.004756	0.
smoke	-0.015486	0.067780	-0.000922	0.005186	0.010354	-0.004756	1.000000	0.
alco	-0.007330	0.067113	0.001408	0.010601	0.035760	0.011246	0.340094	1.
active	-0.035653	-0.016867	-0.000033	0.004780	0.009911	-0.006770	0.025858	0.

In [47]:
```
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix1,annot=True, cmap="Blues")
```

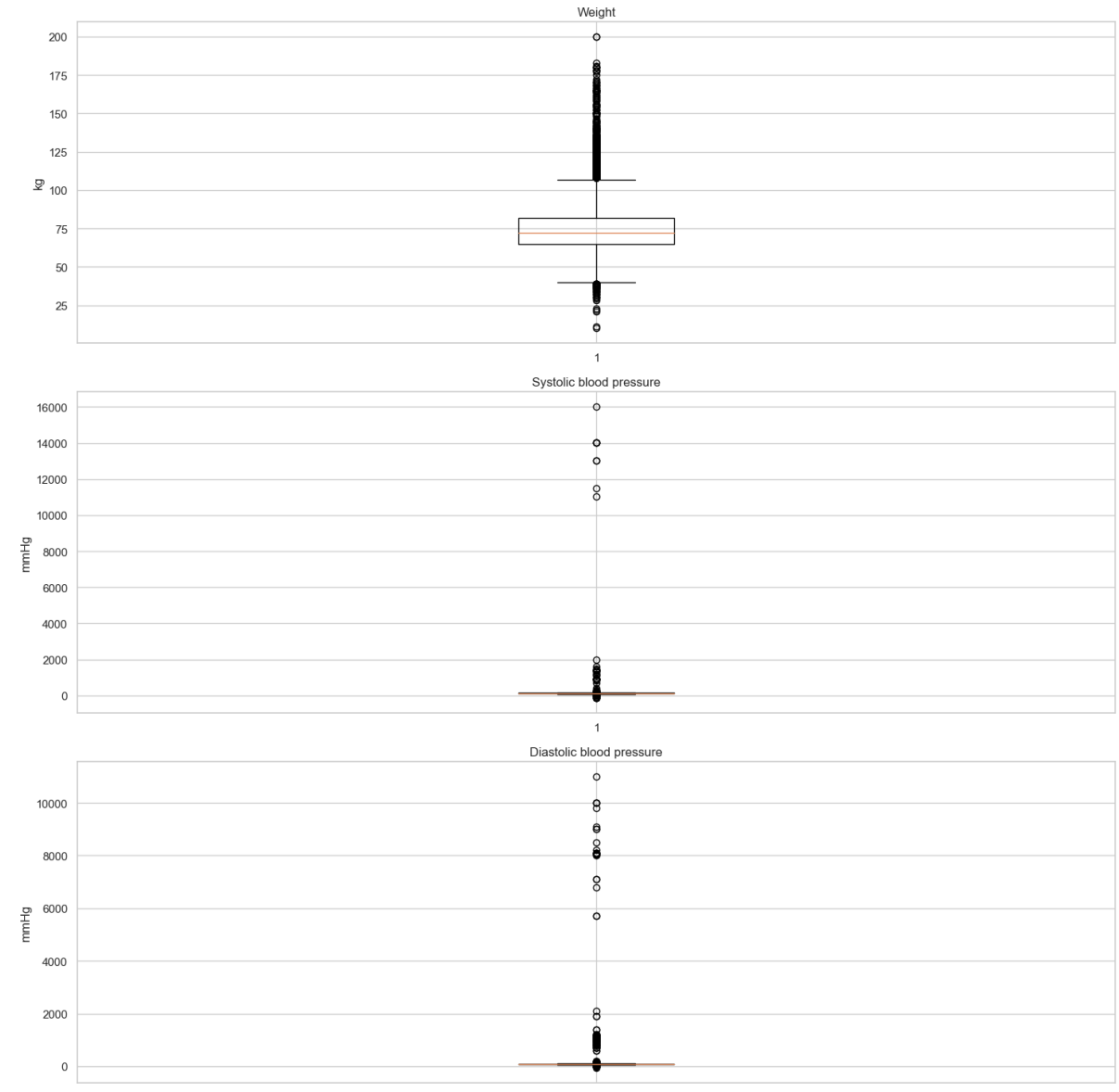
Out[47]: <AxesSubplot:>



We can observe from the pairplots, correlation matrix as well as from the heatmap that there exists no perfect multicollinearity or even high correlation between the variables, indicating that there exists no problem with multicollinearity.

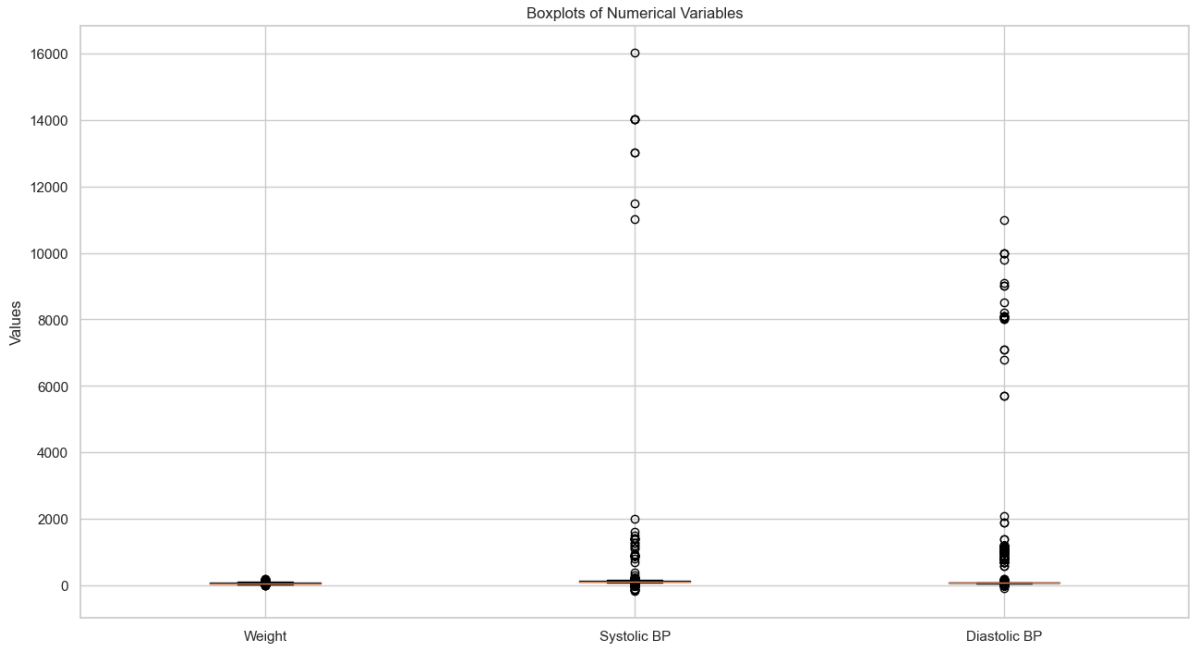
In [49]:
```
```

```
fig, ax = plt.subplots(3, 1, figsize=(15, 15))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac

# Weight
ax[0].boxplot(dt["weight"])
ax[0].set_title("Weight")
ax[0].set_ylabel("kg")

# Systolic Blood Pressure (ap_hi)
ax[1].boxplot(dt["ap_hi"])
ax[1].set_title("Systolic blood pressure")
ax[1].set_ylabel("mmHg")

# Diastolic Blood Pressure (ap_lo)
ax[2].boxplot(dt["ap_lo"])
ax[2].set_title("Diastolic blood pressure")
ax[2].set_ylabel("mmHg")

plt.tight_layout()
plt.show()
```



1

In [51]:
```
fig, ax = plt.subplots(figsize=(15, 8))
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspac

boxplot_data = [dt["weight"], dt["ap_hi"], dt["ap_lo"]]
ax.boxplot(boxplot_data)
ax.set_xticklabels(['Weight', 'Systolic BP', 'Diastolic BP'])
ax.set_ylabel("Values")

plt.title('Boxplots of Numerical Variables')
plt.show()
```



We see that there is an overlap between the three variables, weight, Systolic BP and Diastolic BP. The spread for Susytolic and Diastolic BP is more as compared to weight. It also indicates that weight , Systolic and Diastolic BP are left skewed and have potential outliers. The median for weight is more than for the Systolic and Diastolic BP. The Inter Quartile Range is also larger for the weight variable, showing that the spread for weight is more than for the Systolic and Diastolic BP.

Linear Probability Model

In [76]:
```
modelx = smf.ols(formula='cardio ~  weight + ap_hi + ap_lo+ C(chole
resultsx = modelx.fit(cov_type='HC3')
```

In [77]:
```
resultsx.summary()
```

Out[77]:

OLS Regression Results

Dep. Variable:	cardio	**R-squared:**	0.080
Model:	OLS	**Adj. R-squared:**	0.080
Method:	Least Squares	**F-statistic:**	692.8
Date:	Wed, 06 Dec 2023	**Prob (F-statistic):**	0.00
Time:	03:26:51	**Log-Likelihood:**	-47881.
No. Observations:	70000	**AIC:**	9.578e+04
Df Residuals:	69989	**BIC:**	9.589e+04
Df Model:	10		
Covariance Type:	HC3		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	0.0651	0.011	5.730	0.000	0.043	0.087
C(cholesterol)[T.2]	0.1377	0.006	24.496	0.000	0.127	0.149
C(cholesterol)[T.3]	0.3182	0.006	54.586	0.000	0.307	0.330
C(gluc)[T.2]	0.0229	0.007	3.158	0.002	0.009	0.037
C(gluc)[T.3]	-0.0622	0.008	-8.179	0.000	-0.077	-0.047
C(smoke)[T.1]	-0.0382	0.007	-5.692	0.000	-0.051	-0.025
C(alco)[T.1]	-0.0389	0.008	-4.622	0.000	-0.055	-0.022
C(active)[T.1]	-0.0435	0.005	-9.445	0.000	-0.052	-0.034
weight	0.0053	0.000	41.203	0.000	0.005	0.006
ap_hi	0.0001	4.81e-05	2.986	0.003	4.93e-05	0.000
ap_lo	0.0001	2.12e-05	6.764	0.000	0.000	0.000

Omnibus:	309393.406	**Durbin-Watson:**	1.986
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8207.113
Skew:	0.032	**Prob(JB):**	0.00
Kurtosis:	1.324	**Cond. No.**	1.41e+03

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)
[2] The condition number is large, 1.41e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Probit model

In [86]:
```
modely = smf.probit(
    formula='cardio ~ weight + ap_hi + ap_lo+ C(cholesterol) + C(gl
)
resultsy = modely.fit(disp=0)
```

In [79]:
```
resultsy.summary()
```

Out[79]:

Probit Regression Results

Dep. Variable:	cardio	**No. Observations:**	70000
Model:	Probit	**Df Residuals:**	69989
Method:	MLE	**Df Model:**	10
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	nan
Time:	03:27:36	**Log-Likelihood:**	nan
converged:	True	**LL-Null:**	-48520.
Covariance Type:	nonrobust	**LLR p-value:**	nan

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	nan	nan	nan	nan	nan	nan
C(cholesterol)[T.2]	nan	nan	nan	nan	nan	nan
C(cholesterol)[T.3]	nan	nan	nan	nan	nan	nan
C(gluc)[T.2]	nan	nan	nan	nan	nan	nan
C(gluc)[T.3]	nan	nan	nan	nan	nan	nan
C(smoke)[T.1]	nan	nan	nan	nan	nan	nan
C(alco)[T.1]	nan	nan	nan	nan	nan	nan
C(active)[T.1]	nan	nan	nan	nan	nan	nan
weight	nan	nan	nan	nan	nan	nan
ap_hi	nan	nan	nan	nan	nan	nan
ap_lo	nan	nan	nan	nan	nan	nan

In [88]:
```
import statsmodels.api as sm


X = dt[["weight", "ap_hi", "ap_lo", "cholesterol", "gluc", "smoke",
y = dt["cardio"]

modeln = sm.Probit(y, X).fit(method='bfgs', maxiter=1000)
```

```
print(modeln.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.670835
         Iterations: 28
         Function evaluations: 35
         Gradient evaluations: 35
                          Probit Regression Results
==============================================================================
Dep. Variable:                 cardio   No. Observations:
70000
Model:                         Probit   Df Residuals:
69992
Method:                           MLE   Df Model:
7
Date:                Wed, 06 Dec 2023   Pseudo R-squ.:
0.03219
Time:                        03:32:05   Log-Likelihood:
-46958.
converged:                       True   LL-Null:
-48520.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.02
5      0.975]
------------------------------------------------------------------------------
weight         -0.0008      0.000     -4.107      0.000      -0.00
1      -0.000
ap_hi           0.0004   5.06e-05      7.301      0.000       0.00
0       0.000
ap_lo           0.0004      3e-05     12.427      0.000       0.00
0       0.000
cholesterol     0.3623      0.008     43.858      0.000       0.34
6       0.379
gluc           -0.1538      0.009    -16.345      0.000      -0.17
2      -0.135
smoke          -0.0965      0.018     -5.337      0.000      -0.13
2      -0.061
alco           -0.0643      0.023     -2.825      0.005      -0.10
9      -0.020
active         -0.3364      0.011    -29.512      0.000      -0.35
9      -0.314
==============================================================================
```

In [67]:
```
# log likelihood value:
print(f'results_probit.llf: {modeln.llf}\n')
```

```
results_probit.llf: -46958.478118725034
```

In [68]:
```
# McFadden's pseudo R2:
print(f'results_probit.prsquared: {modeln.prsquared}\n')
```

```
results_probit.prsquared: 0.032188841395919154
```

In [89]:
```
modeln.aic
```

Out[89]:
```
93932.95623745007
```

In [90]:
```
modeln.bic
```

Out[90]:
```
94006.20624161832
```

Logit Model

In [80]:
```
modelz = smf.logit(
    formula='cardio ~ weight + ap_hi + ap_lo+ C(cholesterol) + C(gl
).fit(disp=0)
modelz.summary()
```

Out[80]:

Logit Regression Results

Dep. Variable:	cardio	**No. Observations:**	70000
Model:	Logit	**Df Residuals:**	69989
Method:	MLE	**Df Model:**	10
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.1271
Time:	03:28:29	**Log-Likelihood:**	-42351.
converged:	True	**LL-Null:**	-48520.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-6.4584	0.082	-79.125	0.000	-6.618	-6.298
C(cholesterol)[T.2]	0.4369	0.026	17.085	0.000	0.387	0.487
C(cholesterol)[T.3]	1.2517	0.034	36.865	0.000	1.185	1.318
C(gluc)[T.2]	0.0483	0.034	1.425	0.154	-0.018	0.115
C(gluc)[T.3]	-0.2923	0.038	-7.745	0.000	-0.366	-0.218
C(smoke)[T.1]	-0.2049	0.031	-6.545	0.000	-0.266	-0.144
C(alco)[T.1]	-0.2024	0.040	-5.098	0.000	-0.280	-0.125
C(active)[T.1]	-0.2113	0.021	-10.187	0.000	-0.252	-0.171
weight	0.0136	0.001	21.951	0.000	0.012	0.015
ap_hi	0.0431	0.001	71.977	0.000	0.042	0.044
ap_lo	0.0003	6.51e-05	4.068	0.000	0.000	0.000

In [81]:
```
# log likelihood value:
print(f'results_logit.llf: {modelz.llf}\n')
```

```
results_logit.llf: -42351.00751529075
```

In [82]:
```
# McFadden's pseudo R2:
print(f'results_logit.prsquared: {modelz.prsquared}\n')
```

```
results_logit.prsquared: 0.12714850880146011
```

In [91]:
```
modelz.aic
```

Out[91]: 84724.0150305815

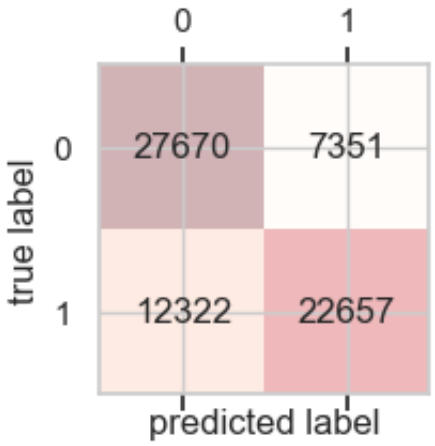
In [92]:
```
modelz.bic
```

Out[92]: 84824.73378631285

Confusion Matrix

In [83]:
```
#for linear probability
X_cols = ["weight", "ap_hi", "ap_lo", "cholesterol", "gluc", "smoke
lr = LogisticRegression()
logit_mod = lr.fit(dt[X_cols], dt['cardio'])
conf_mat = confusion_matrix(dt['cardio'], lr.predict(dt[X_cols]))
print(conf_mat)
print('Accuracy =', lr.score(dt[X_cols],dt['cardio']))

# Confusion matrix plot Raschka (2014)
fig, ax = plt.subplots(figsize=(2, 2))
ax.matshow(conf_mat, cmap=plt.cm.Reds, alpha=0.3)
for i in range(conf_mat.shape[0]):
    for j in range(conf_mat.shape[1]):
        ax.text(x=j, y=i,
        s=conf_mat[i, j],
        va='center', ha='center')
plt.xlabel('predicted label')
plt.ylabel('true label')
plt.show()
```

```
[[27670  7351]
 [12322 22657]]
Accuracy = 0.7189571428571429
```



True Positives (TP): 22,657 cases were correctly predicted as positive (e.g., predicted as having cardiovascular disease). False Positives (FP): 7,351 cases were wrongly predicted as positive (predicted as having cardiovascular disease, but they do not). True Negatives (TN): 27,670 cases were correctly predicted as negative (e.g., predicted as not having cardiovascular disease). False Negatives (FN): 12,322 cases were wrongly predicted as negative (predicted as not having cardiovascular disease, but they do). The logit model gives an accruacy level of approximately 72%, indicating that it correctly predcited outcomes for 71.90% of the cases. Since there are high TP and TN values, it predcited high number of true negative and positive cases. There are also moderate number of false positive and negative cases but these are not overwhelming.

In the aobve assessment of running our model with "cardio" as the dependent variable and weight, systolic blood pressue, cholesterol, glucose, smoke, alcohol and activity as the dependent variables, we see a similar result for all three. There are some slight changes across the statistical summary fo each model. Overall on analysis, I observe that the linear probabaility model shows the lowest AIC and BIC value amongst the three models. When looking at the adjusted R squared, it is 8% and the psuedo R sqaured is higher for the probit model. When looking at the log likelihood value it is the logit model giving the goodness of fit. Looking at all these parameters, and observing that association to the probability of outcomes is almost the same across models, we will choose the logit model, for higher psuedo R sqaured, and highest Log Likelihood value. Even though the AIC and BIC values are the lowest for the linear probability model, but with several reasons supporting logit, I will still choose the Logit model.

In [117]:
```
X_new = pd.DataFrame({
    'weight': [74, 65, 200, 82],
    'ap_hi': [128, 140, 130, 150],
    'ap_lo': [80, 90, 60, 85],
    'cholesterol': [1, 2, 3, 0],
    'gluc': [1, 2, 3, 0],
    'smoke': [0, 1, 0, 0],
    'alco': [0, 1, 0, 0],
    'active': [0, 1, 0, 0]
})
X_new
```

Out[117]:

	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	74	128	80	1	1	0	0	0
1	65	140	90	2	2	1	1	1
2	200	130	60	3	3	0	0	0
3	82	150	85	0	0	0	0	0

In [120]:
```
X_new=X_new.dropna()
X_new
```

Out[120]:

	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	74	128	80	1	1	0	0	0
1	65	140	90	2	2	1	1	1
2	200	130	60	3	3	0	0	0
3	82	150	85	0	0	0	0	0

In [122]:
```
categorical_cols = ['cholesterol', 'gluc', 'smoke', 'alco', 'active
for col in categorical_cols:
    X_new[col] = pd.Categorical(X_new[col], categories=[0, 1, 2, 3]
print(X_new.dtypes)
```

```
weight            int64
ap_hi             int64
ap_lo             int64
cholesterol    category
gluc           category
smoke          category
alco           category
active         category
dtype: object
```

In [123]:
```
print(X_new.iloc[2])
```

```
weight         200
ap_hi          130
ap_lo           60
cholesterol      3
gluc             3
smoke            0
alco             0
active           0
Name: 2, dtype: int64
```

In [125]:
```
print(X_new['cholesterol'].unique())
print(X_new['gluc'].unique())
print(X_new['smoke'].unique())
print(X_new['alco'].unique())
print(X_new['active'].unique())
```

```
[1, 2, 3, NaN]
Categories (4, int64): [0, 1, 2, 3]
[1, 2, 3, NaN]
Categories (4, int64): [0, 1, 2, 3]
[0, 1]
Categories (4, int64): [0, 1, 2, 3]
[0, 1]
Categories (4, int64): [0, 1, 2, 3]
[0, 1]
Categories (4, int64): [0, 1, 2, 3]
```

In [132]:
```
X_new['cholesterol'] = pd.Categorical(X_new['cholesterol'], categor
X_new['gluc'] = pd.Categorical(X_new['gluc'], categories=[1, 2, 3])

X_new['cholesterol'].fillna(1, inplace=True)
X_new['gluc'].fillna(1, inplace=True)

print(X_new['cholesterol'].unique())
print(X_new['gluc'].unique())
```

```
[1, 2]
Categories (3, int64): [1, 2, 3]
[1, 2]
Categories (3, int64): [1, 2, 3]
```

In [133]:
```
predictions_logit = modelz.predict(X_new)
print(f'predictions_logit: \n{predictions_logit}\n')
```

```
predictions_logit:
0    0.523709
1    0.588806
2    0.874733
3    0.760341
dtype: float64
```

For observations 0 and 1, there are around 52 - 59% chances of cardiovasular disease presence when the weight of the person is around 74 to 65 kg, systolic and diastolic pressure 128-140 and 80-90 respectively, and their cholesterol and glucose are normal or above normal. The only set of difference occurs in their lifestyle, where in the first observation the person doesn't smoke, drink or is not active but in the second observations he smokes, drinks and is active. In the third set of observations, the chances of cardiovascular disease is accounting to a very high probability of 87% wherein the weight is as high as 200kg, systolic an diastolic blood pressure is 130 and 60 respectively, cholesterol and glusoces are well above normal levels but the person doesn't smoke, drink or is active. This indicates a very important observation, that obesity, high cholesterol levels and diabetes are major causes of cardiovascular diseases and the interplay of these alone can siginificantly increase chances of cardiovascular health problems. In the last prediction, there are 76% chances of cardiovascular disease presence, wherein the weight if 82kg, and the systolic blood pressure is as high as 150 and diastolic is normal ranging at 85, and the person doesn't have cholesterol, diabetes and doesn't smoke, drink or is active. We observe that obesity and high blood pressure can itself lead to larger probabaility of cardiovascular diseases in people. This analysis gives a very clear picture on the major causes and how the interplay of different co-morbidities and lifestyle choices can effect you chances of cardiovascular diseases. The main cause can be attributed to weight and and blood pressure levels, seeing the difference in result from predictions in observation 0 and 1 to predictions in observation 2 and 3. This also gives a intuitively reliable predcitions for the values that were given to the model.