Statistical Methods Project
MA541

Title: Statistical Analysis on Car MPG data

Guide: Prof. Hadi Safari

Author: Siddhesh More

Abstract/Introduction:
In this project we will do many statistical hypothesis testing, apply estimation statistics and interpret the results we get. We will also validate this with the findings from EDA. We will apply both parametric and non-parametric tests. We will report all the important insights we get. Testing data with different hypothesis and taking out the insights from the observed results.
We will be performing following steps in order to do the statistical analysis:

1. Preprocess the data
2. Tests for Independence between two categorical attributes
3. Normality test for numeric attributes
4. Correlation between numeric attributes
5. Parametric and non-parametric test for samples
6. ANOVA test

The code file contains EDA along with the statistical and numerical analysis.

Data description:
The data is selected from UCI repository. The link for the dataset is here.
Information regarding data
Title: Auto-Mpg Data
Number of Instances: 398
Number of Attributes: 9 including the class attribute
Attribute Information:

1. mpg: continuous

2. cylinders: multi-valued discrete

3. displacement: continuous

4. horsepower: continuous

5. weight: continuous

6. acceleration: continuous

7. model year: multi-valued discrete

8. origin: multi-valued discrete

9. car name: string (unique for each instance)

All the attributes are self-explanatory.

This data is not complex and is good for analysis as it has a nice blend of both categorical and numerical attributes.

Insights of the dataset

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

Methodology/Analysis and results:
Before starting to analyze the data we will be preprocessing the data. The data had some null values and duplicate rows. Foe our convenience for analysis we have dropped these duplicate rows and rows having null values. After the preprocessing we are left with 392 observations so we will perform all the analysis on this clean data.

Statistical Hypothesis tests
Statistical Hypothesis Tests are a very simple and straightforward concept. We begin by making assumptions about the data, such as that the two samples have the same mean. Then we calculate the probability of seeing the supplied data if this assumption is correct. We reject the assumption if the likelihood is near to zero, and we fail to reject the assumption if the likelihood is larger than some threshold (specified by us).
The assumption is known as Hypothesis in statistics, the likelihood we obtain is known as p-value, the threshold we choose is either level of significance or critical value, and the test we use is known as Statistical Hypothesis Tests.
So, if the likelihood we obtain is extremely near to zero, it means that if our hypothesis is correct, the possibility of observing/occurring this data is very low, implying that our assumption is incorrect. So, in the case of means of two samples, if the resulting p-value is extremely near to zero, we may conclude that assuming the two samples have the same mean, the data we have in hand is very unlikely to have created, and so we reject our assumption.

The **statistical significance** of any finding is done by intrerpreting the p-values. P-value tells us that whether are findings are due to same real change or they are just random fluctuations.

- p-value ≤ $\alpha\alpha$: significant result, reject null hypothesis.
- p-value > $\alpha\alpha$: not significant result, fail to reject the null hypothesis.

A p-value can be calculated from a test statistic by retrieving the probability from the test statistics cumulative density function (CDF).

Tests for independence between two categorical variables

We are setting value of alpha to 0.05.

Pearson's Chi-square test

The Chi-square statistic is a **non-parametric statistic** tool designed to analyze group differences when the dependent variable is measured at a nominal level(ordinal data can also be used). It is commonly used to compare observed data with data we would expect to obtain according to a specific hypothesis.

Fisher's exact test

Fisher's exact test is used to determine whether there is a significant association between two categorical variables in a contingency table. Fisher's exact test is an alternative to Pearson's chi-squared test for independence. While actually valid for all sample sizes, Fisher's exact test is practically applied when sample sizes are small. A general recommendation is to use Fisher's exact test- instead of the chi-squared test - whenever more than 20 % of cells in a contingency table have expected frequencies < 5.

```
#Contingency Table
pd.crosstab(df.origin, df.model_year)
```

model_year	70	71	72	73	74	75	76	77	78	79	80	81	82
origin													
europe	5	4	5	7	6	6	8	4	6	4	8	3	2
japan	2	4	5	4	6	4	4	6	8	2	13	12	9
usa	22	19	18	29	14	20	22	18	22	23	6	13	19

So chi2 assumption failed for every pair but it's not that we can't apply, we can but the results are not reliable. But the contingency table of origin and model_year is still good to try most values are >= 5.

H_0: origin are model_year are independent. α=0.05

```
chi2, p, dof, expected_values = stats.chi2_contingency(observed_values)
chi2, p, dof, expected_values
```

```
(123.76491109767298,
 8.381476294026467e-26,
 4,
 array([[ 16.47959184,  15.95918367,  35.56122449],
        [ 19.14540816,  18.54081633,  41.31377551],
        [ 59.375     ,  57.5       , 128.125     ]]))
```

```
if p <= ALPHA:
```

```
    print(f'Rejected H0 under significance level {ALPHA} `origin` & `model
_year` are dependent.')
else:
    print(f'Fail to reject H0 due to lack of evidence under significance l
evel {ALPHA} `origin` & `model_year` are independent.')
```

```
Rejected H0 under significance level 0.05 `origin` & `model_year` are depe
ndent.
```

Let's test dependency of all categorical attributes with `mpg_level`.

```
df_cat_label =  pd.concat([df.loc[:, ['origin', 'mpg_level']].apply(lambda
x: LabelEncoder().fit_transform(x)),
                           df.loc[: , 'cylinders': 'model_year']], axis=1)

df_cat_label.head()
```

	origin	mpg_level	cylinders	model_year
0	2	2	8	70
1	2	1	8	70
2	2	2	8	70
3	2	1	8	70
4	2	2	8	70

```
chi2_res = feature_selection.chi2(df_cat_label, df.mpg_level)
df_chi2 = pd.DataFrame({
    'attr1': 'mpg_level',
    'attr2': df_cat_label.columns,
    'chi2': chi2_res[0],
    'p': chi2_res[1],
    'alpha': ALPHA
})
df_chi2['H0'] = df_chi2.p.apply(lambda x: 'rejected' if x <= ALPHA else 'f
ail to reject')
df_chi2['relation'] = df_chi2.H0.apply(lambda x: 'dependent' if x=='reject
ed' else 'independent')
df_chi2
```

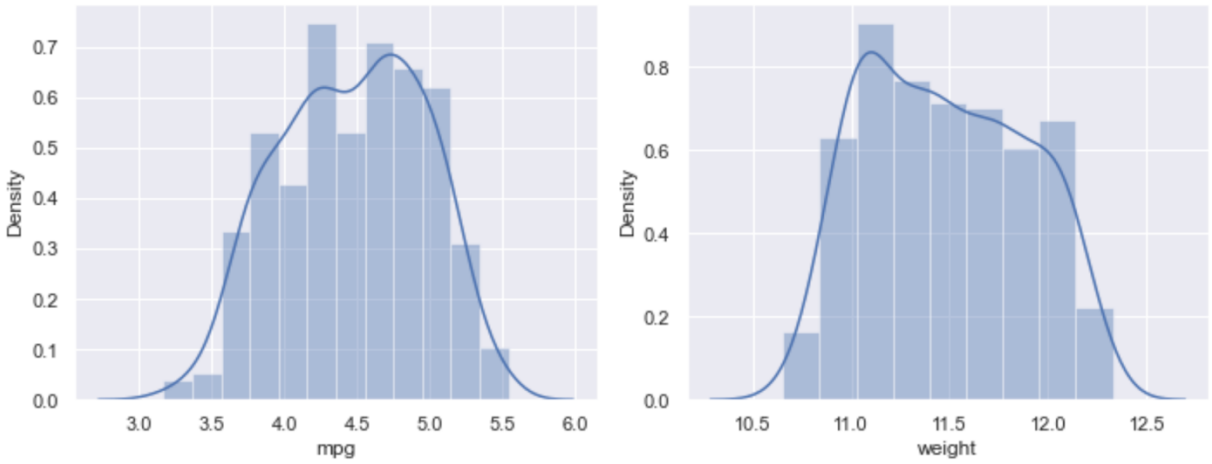
	attr1	attr2	chi2	p	alpha	H0	relation
0	mpg_level	origin	28.395578	6.823049e-07	0.05	rejected	dependent
1	mpg_level	mpg_level	210.159363	2.314591e-46	0.05	rejected	dependent
2	mpg_level	cylinders	127.418999	2.144450e-28	0.05	rejected	dependent
3	mpg_level	model_year	21.742075	1.900065e-05	0.05	rejected	dependent

Statistical Tests for Numerical Attributes

A **log-normal distribution** is a distribution of a random variable whose logarithm is normally distributed. Thus, if the random variable X is log-normally distributed, then Y = ln(X) has a normal distribution.

We will check whether mpg and weight are log-normal or not.

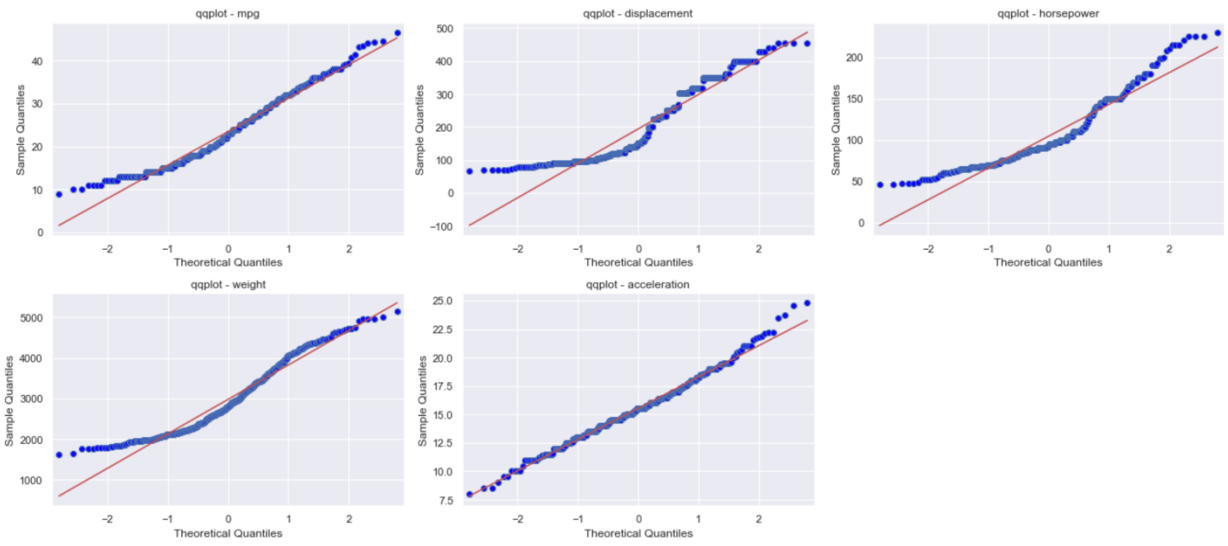
```
fig = pyplot.figure(1, (10, 4))

ax = pyplot.subplot(1,2,1)
sns.distplot(np.log2(df.mpg))
pyplot.tight_layout()

ax = pyplot.subplot(1,2,2)
sns.distplot(np.log2(df.weight))
pyplot.tight_layout()

pyplot.show()
```



So after applying log transformation we find that weight is not log-normal but mpg visually **looks like log-normal**.

Let's check for normality using quantile-quantile plots.

```
# quantile-quantile plots on original data
fig = pyplot.figure(1, (18,8))

for i,num in enumerate(nums):
    ax = pyplot.subplot(2,3,i+1)
    qqplot(df[num], line= 's', ax=ax)
    ax.set_title(f'qqplot - {num}')
    pyplot.tight_layout()
```



so both histplot & qqplot of `acceleration` indicates that it is indeed close to gaussian.

Statistical Normality Tests

We will do hypothesis testing for the normality of numerical attributes using the shapiro wilk test.

H_0: Data is drawn from normal distribution. α=0.05

```
# let's contruct a function
def shapiro_wilk_test(df: pd.DataFrame, cols: list, alpha=0.05):
    # test the null hypothesis for columns given in `cols` of the datafram
e `df` under significance level `alpha`.
    for col in cols:
        _,p = stats.shapiro(df[col])
        if p <= alpha:
            print(f'''\nRejected H0 under significance level {alpha}\n{col
} doesn't seems to be normally distributed''')
        else:
```

```
            print(f'''\nFail to reject H0 due to lack of evidence under si
gnificance level {alpha}\n{col} seem to be normally distributed''')
shapiro_wilk_test(df, nums)
```

```
Rejected H0 under significance level 0.05
mpg doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
displacement doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
horsepower doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
weight doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
acceleration doesn't seems to be normally distributed
```

We expected acceleration to be normally distributed but are test rejected it, let's check the p-value.

```
_, p = stats.shapiro(df.acceleration)
p
```

```
0.03054318018257618
```

So we rejected it under the significance level of 5% but if it was 2.5% then we would have failed to reject the null-hypothesis. We won't change the p-value now otherwise it will be **p-hacking**. One possible reason for the rejection of H_0 is maybe our data is not scaled and I think scaling it helps.
We will now apply power transform to **make the data more gaussian like**. And after that check for normality on the transformed data.

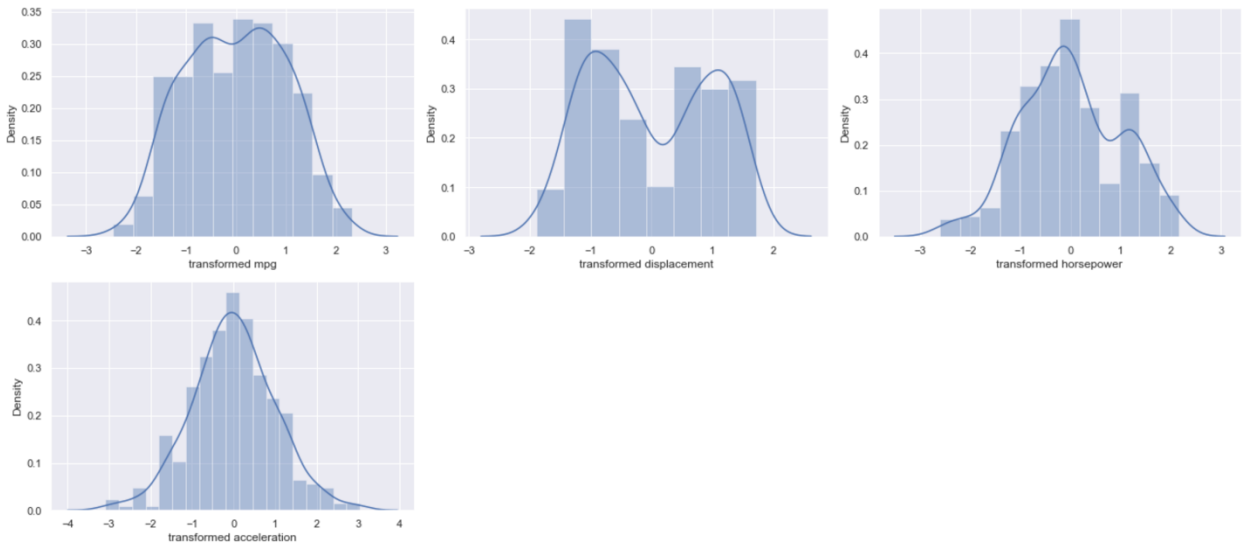
Power Transform: It transforms data featurewise to make it more Gaussian-like. Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired just like here.

```
from sklearn.preprocessing import PowerTransformer

df_tfnum = pd.DataFrame(PowerTransformer().fit_transform(df[nums]), column
s=nums)
df_tfnum.head()
```

	mpg	displacement	horsepower	weight	acceleration
0	-0.635451	1.119336	0.854984	0.0	-1.317572
1	-1.142697	1.324530	1.443754	0.0	-1.522941
2	-0.635451	1.175211	1.216062	0.0	-1.732292
3	-0.965340	1.103648	1.216062	0.0	-1.317572
4	-0.796543	1.093078	1.044925	0.0	-1.945873

On removing nans from the dataframe the power transformer is making the entire weight column 0. I am unable to find the reason for this. So, for now we will leave weight.

```
fig = pyplot.figure(1, (18,8))

for i,num in enumerate(['mpg', 'displacement', 'horsepower', 'acceleration
']):
    ax = pyplot.subplot(2,3,i+1)
    sns.distplot(df_tfnum[num])
    ax.set_xlabel(f'transformed {num}')
    pyplot.tight_layout()
```



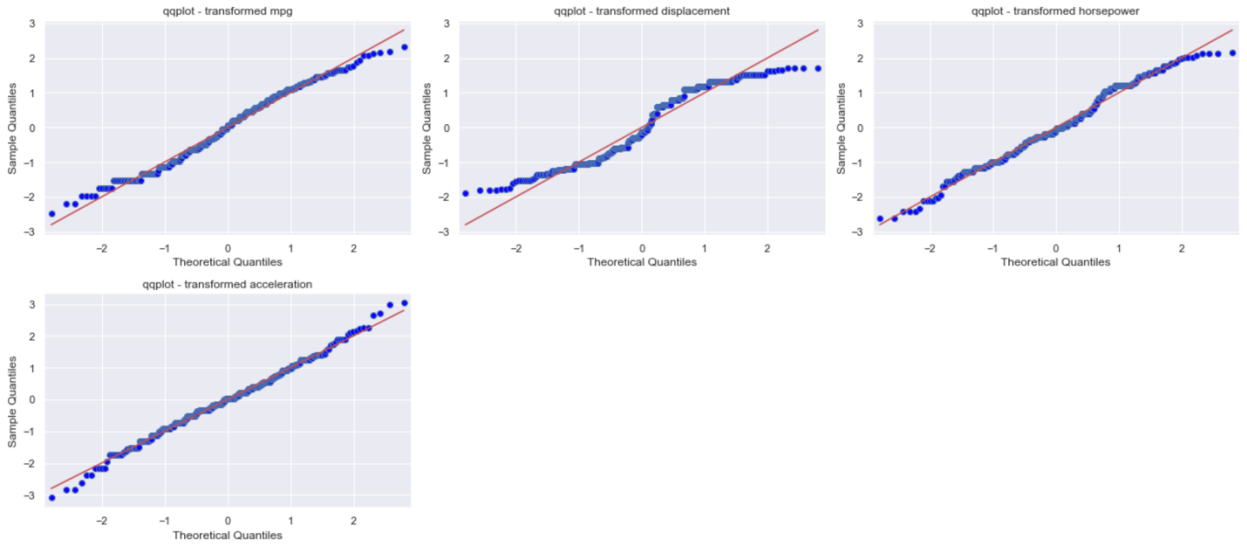
Power transforms does two things, first it scaled the data i.e., now data is centered at 0 and also made the distribution more gaussian-like simultaneously preserving the original structure. It does so by applying transformations like square-root, log etc.

acceleration is still gaussian, skewness is removed from mpg & weight making mpg gaussian-like. Also the distribution for displacement is improved now it's bimodal which respects the observation.

One thing you have noticed that after applying power transform distribution of mpg & weight is quite similar to what we get on applying log transform.

Let's check for normality using quantile-quantile plots.

```
# quantile-quantile plots on transformed data
fig = pyplot.figure(1, (18,8))

for i,num in enumerate(['mpg', 'displacement', 'horsepower', 'acceleration
']):
    ax = pyplot.subplot(2,3,i+1)
    qqplot(df_tfnum[num], line='s', ax=ax)
    ax.set_title(f'qqplot - transformed {num}')
    pyplot.tight_layout()
```




```
shapiro_wilk_test(df_tfnum, ['mpg', 'displacement', 'horsepower', 'acceler
ation'])
```

```
Rejected H0 under significance level 0.05
mpg doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
displacement doesn't seems to be normally distributed

Rejected H0 under significance level 0.05
horsepower doesn't seems to be normally distributed
```

```
Fail to reject H0 due to lack of evidence under significance level 0.05
acceleration seem to be normally distributed
```

```
_, p = stats.shapiro(df_tfnum.acceleration)
p
```

```
0.3864997923374176
```

Indeed, on normalizing the data the likelihood of observing acceleration assuming it's normal is very high as compared to earlier.

So `acceleration` is normally distributed both visually and statistically.

Spearman's Rank Correlation

Spearman's Correlation is a **non-parametric rank correlation** and is also interpretable because $Corr_s(x,y) \in [-1,1]$. In this instead of calculating the coefficient using covariance and standard deviations on the samples themselves, these statistics are calculated by converting the raw data into rank data hence non-parametric. **This is a common approach used in non-parametric statistics.**
As a statistical hypothesis test, the method assumes that the samples are uncorrelated (fail to reject H_0).

So for all correlation test b/w mpg and other attribute our null hypothesis will be,

H_0: mpg and other attribute are not correlated. α=0.05

```
for num in nums:
    if num == 'mpg':
        continue

    corr, p = stats.spearmanr(df.mpg, df[num])

    print(f'\n* `mpg` & `{num}`\n')
    print(f'corr: {round(corr, 4)} \t p: {p}')

    if p <= ALPHA:
        print(f'Rejected H0 under significance level {ALPHA}, mpg & {num}
are correlated')
    else:
        print(f'''Fail to reject H0 due to lack of evidence under signific
ance level {ALPHA},
              mpg & {num} are not correlated''')
```

```
* `mpg` & `displacement`
```

```
corr: -0.8552   p: 2.1957775993226176e-113
Rejected H0 under significance level 0.05, mpg & displacement are correlat
ed
```

* `mpg` & `horsepower`

```
corr: -0.8536   p: 1.619383245501938e-112
Rejected H0 under significance level 0.05, mpg & horsepower are correlated
```

* `mpg` & `weight`

```
corr: -0.8756   p: 2.662377938025222e-125
Rejected H0 under significance level 0.05, mpg & weight are correlated
```

* `mpg` & `acceleration`

```
corr: 0.4415    p: 3.9036035663531793e-20
Rejected H0 under significance level 0.05, mpg & acceleration are correlat
ed
```

So all the H0 are rejected under the significance level of 5%. Accept `acceleration` all the other c orrelations are very high and this is also evident from our previous plots.

We now create a dataframe for the correlation b/w every pair.

```python
def test_correlation(x1, x2, method='spearman', alpha=0.05):
    # this function returns correlation, p-value and H0 for `x1` & `x2`

    ALLOWED_METHODS = ['pearson', 'spearman', 'kendall']
    if method not in ALLOWED_METHODS:
        raise ValueError(f'allowed methods are {ALLOWED_METHODS}')

    if method=='pearson':
        corr, p = stats.pearsonr(x1,x2)
    elif method=='spearman':
        corr, p = stats.spearmanr(x1,x2)
    else:
        corr, p = stats.kendalltau(x1,x2)

    h0 = (
    'rejected'
    if p<=ALPHA else
    'fail to reject')
```

```
    return corr, p, h0
```

```
df_corr = pd.DataFrame(columns=['attr1', 'attr2', 'corr', 'p', 'H0'])

for combo in itertools.combinations(nums, r=2):
    corr, p, h0 = test_correlation(df[combo[0]], df[combo[1]])
    df_corr = df_corr.append({'attr1':combo[0], 'attr2':combo[1],
                              'corr':round(corr, 5), 'p':p, 'H0':h0}, igno
re_index=True)

df_corr
```

	attr1	attr2	corr	p	H0
0	mpg	displacement	-0.85523	2.195778e-113	rejected
1	mpg	horsepower	-0.85362	1.619383e-112	rejected
2	mpg	weight	-0.87559	2.662378e-125	rejected
3	mpg	acceleration	0.44154	3.903604e-20	rejected
4	displacement	horsepower	0.87617	1.126737e-125	rejected
5	displacement	weight	0.94563	2.463170e-192	rejected
6	displacement	acceleration	-0.49940	4.061210e-26	rejected
7	horsepower	weight	0.87882	2.182674e-127	rejected
8	horsepower	acceleration	-0.65814	5.157840e-50	rejected
9	weight	acceleration	-0.40511	6.484246e-17	rejected

Correlation of pairs (mpg, acceleration), (displacement, acceleration) and (weight, acceleration) is moderate whereas remaining all pairs has very high correlation between them.

We will now test whether two samples has the same mean or not. For this we have two types of significance tests for two different conditions.

Parametric Statistical Significance Tests

1. **t-test** - It tests whether the two independent normal distributed samples has the same mean or not.
2. **Analysis of Variance Test (ANOVA)** - It tests whether the two or more independent normal distributed samples has the same mean or not.

ANOVA is same as t-test but for more than 2 variables. So either we can apply t-test pair-wise of apply ANOVA once. Also ANOVA only tells whether all samples are same or not, it doesn't quantify which samples differ or by how much.

Non-Parametric Statistical Significance Tests

1. **Mann-Whitney U Test** - Non-parameetric equivalent of Student's t-test.
2. **Kruskal-Wallis H** - Non-parameetric equivalent of ANOVA (it's for median).

We will apply appropriate test depending on the sample, i.e., if samples are normally distributed then parametric tests otherwise non-parametric tests.

Let's test whether acceleration in `japan` and `usa` has the same mean.

First we check whether acceleration of both japan and usa are normally distributed or not and then apply the applicable tests.

```
shapiro_wilk_test(df[df.origin=='japan'], ['acceleration'])
```

```
Fail to reject H0 due to lack of evidence under significance level 0.05
acceleration seem to be normally distributed
```

```
shapiro_wilk_test(df[df.origin=='usa'], ['acceleration'])
```

```
Fail to reject H0 due to lack of evidence under significance level 0.05
acceleration seem to be normally distributed
```

So both are normally distributed so we can apply parametric test.

H_0: acceleration of japan and acceleration of usa has same sample mean. α=0.05

```
# because the variance is not same for the two distributions hence equal_var=False
_, p = stats.ttest_ind(df[df.origin=='japan'].acceleration, df[df.origin=='usa'].acceleration, equal_var=False)

if p <= ALPHA:
    print(f'Rejected H0 under {ALPHA*100}% significance, Different distributions.')
else:
    print(f'Fail to Reject H0 under {ALPHA*100}% significance, Same distributions.')
```

```
Rejected H0 under 5.0% significance, Different distributions.
```

```
_, p = stats.f_oneway(df[df.origin=='japan'].acceleration, df[df.origin=='
usa'].acceleration, df[df.origin=='europe'].acceleration)

if p <= ALPHA:
    print(f'Rejected H0 under {ALPHA*100}% significance, Different distrib
utions.')
else:
    print(f'Fail to Reject H0 under {ALPHA*100}% significance, Same distri
butions.')
```

```
Rejected H0 under 5.0% significance, Different distributions.
```

Let's test whether horsepower across all the regions has the same distribution or not.

```
shapiro_wilk_test(df[df.origin=='japan'], ['horsepower'])
```

```
Rejected H0 under significance level 0.05
horsepower doesn't seems to be normally distributed
```

```
shapiro_wilk_test(df[df.origin=='europe'], ['horsepower'])
```

```
Rejected H0 under significance level 0.05
horsepower doesn't seems to be normally distributed
```

```
shapiro_wilk_test(df[df.origin=='usa'], ['horsepower'])
```

```
Rejected H0 under significance level 0.05
horsepower doesn't seems to be normally distributed
```

So all of them are not normally distributed so we will apply non-parametric test.

H_0: Sample distributions are equal for horsepower across region. α=0.05

```
_, p = stats.kruskal(df[df.origin=='japan'].horsepower, df[df.origin=='usa
'].horsepower, df[df.origin=='europe'].horsepower)

if p <= ALPHA:
    print(f'Rejected H0 under {ALPHA*100}% significance, Different distrib
utions.')
else:
    print(f'Fail to Reject H0 under {ALPHA*100}% significance, Same distri
butions.')
```

```
Rejected H0 under 5.0% significance, Different distributions.
```

Test whether acceleration has same distribution for samples with mpg_level high & medium.

```
_, p = stats.mannwhitneyu(df[df.mpg_level=='high'].acceleration, df[df.mpg
_level=='medium'].acceleration)

if p <= ALPHA:
    print(f'Rejected H0 under {ALPHA*100}% significance, Different distrib
utions.')
else:
    print(f'Fail to Reject H0 under {ALPHA*100}% significance, Same distri
butions.')
```

```
Rejected H0 under 5.0% significance, Different distributions.
```

Test for mpg distribution across the years.

```
acc_gb_year = df.groupby('model_year')['mpg']

acc_yr = []
for yr in df.model_year.unique():
    acc_yr.append(list(acc_gb_year.get_group(yr)))
```

```
_, p = stats.kruskal(*acc_yr)

if p <= ALPHA:
    print(f'Rejected H0 under {ALPHA*100}% significance, Different distrib
utions.')
else:
    print(f'Fail to Reject H0 under {ALPHA*100}% significance, Same distri
butions.')
```

```
Rejected H0 under 5.0% significance, Different distributions.
```

ANOVA test

Relation between Categorical and Continous attributes

```
result_f = feature_selection.f_classif(df.loc[:, 'mpg': 'acceleration'], d
f.cylinders)
anova_test_cat = pd.DataFrame({
    'cat-attr': 'cylinders',
    'cont-attr': df.loc[:, 'mpg': 'acceleration'].columns,
    'f': result_f[0],
    'p': result_f[1],
    'alpha': ALPHA
```

```
})
anova_test_cat['H0'] = anova_test_cat.p.apply(lambda x: 'rejected' if x <=
ALPHA else 'fail to reject')
anova_test_cat['relation'] = anova_test_cat.H0.apply(lambda x: 'dependent'
if x=='rejected' else 'independent')
anova_test_cat
```

	cat-attr	cont-attr	f	p	alpha	H0	relation
0	cylinders	mpg	172.954629	8.785541e-85	0.05	rejected	dependent
1	cylinders	displacement	933.260216	3.054493e-197	0.05	rejected	dependent
2	cylinders	horsepower	297.855648	1.079822e-116	0.05	rejected	dependent
3	cylinders	weight	414.851887	1.751845e-138	0.05	rejected	dependent
4	cylinders	acceleration	48.677596	3.709542e-33	0.05	rejected	dependent

```
result_f = feature_selection.f_classif(df_cat_label[['origin', 'cylinders'
, 'model_year']], df.mpg)
anova_test_cat = pd.DataFrame({
    'cont-attr': 'mpg',
    'cat-attr': ['origin', 'cylinders', 'model_year'],
    'f': result_f[0],
    'p': result_f[1],
    'alpha': ALPHA
})
anova_test_cat['H0'] = anova_test_cat.p.apply(lambda x: 'rejected' if x <=
ALPHA else 'fail to reject')
anova_test_cat['relation'] = anova_test_cat.H0.apply(lambda x: 'dependent'
if x=='rejected' else 'independent')
anova_test_cat
```

	cont-attr	cat-attr	f	p	alpha	H0	relation
0	mpg	origin	2.345244	3.516516e-09	0.05	rejected	dependent
1	mpg	cylinders	12.104269	2.855307e-63	0.05	rejected	dependent
2	mpg	model_year	3.795294	3.748099e-20	0.05	rejected	dependent

Conclusion:
Hence, we have performed various tests on the data like Chi-square test, Normality tests, Spearman rank correlation test, and ANOVA test. Observations and conclusions from all these tests are stated just after the test has performed.

References:

1. Guidance: Prof. Hadi Safari
2. Source of the dataset: https://archive.ics.uci.edu/ml/datasets/auto+mpg