In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# read csv into dataframe, categorical data already in encoded labels
df = pd.read_csv('data.csv')
df.head(5)
```

Out[ ]:

	Purchase	SUS	Duration	Gender	ASR_Error	Intent_Error
0	1	84	254	0	3	2
1	0	58	247	0	6	9
2	0	56	125	1	6	8
3	0	55	22	0	11	7
4	1	95	262	0	2	3

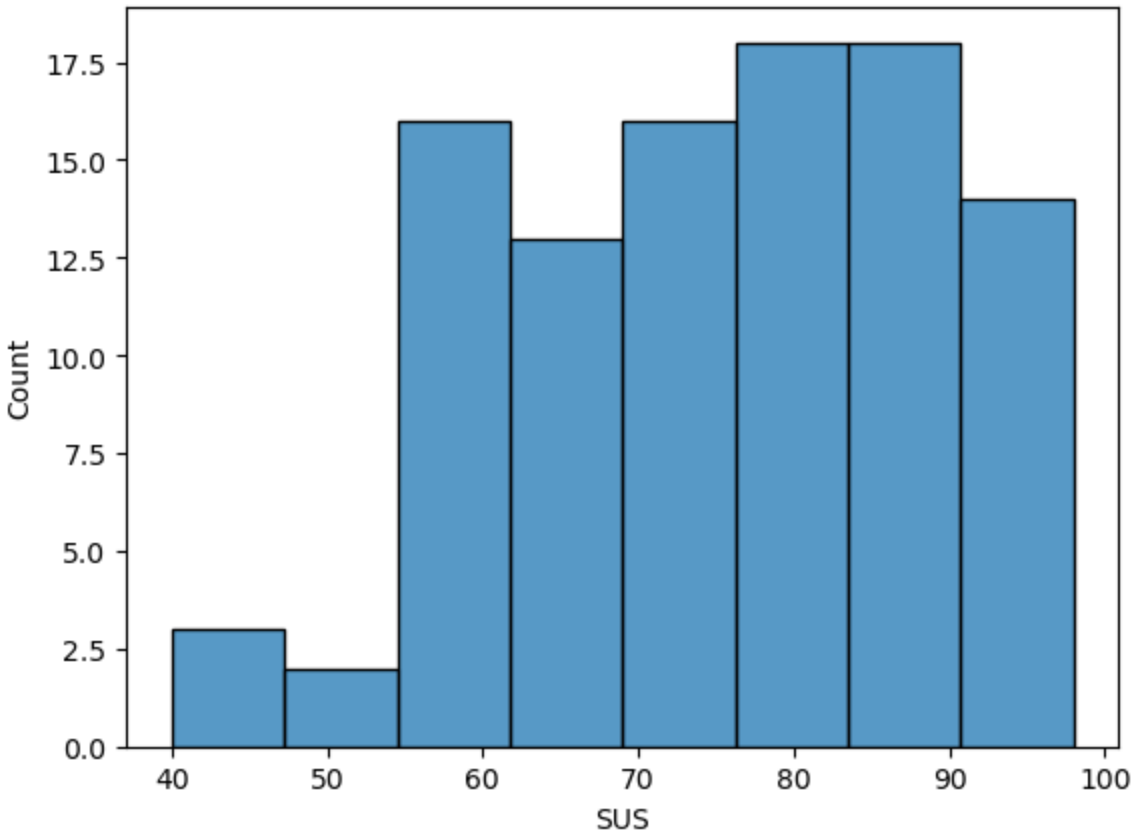
In [ ]:
```
# checking for missing values
df.isna().sum()
```

Out[ ]:
```
Purchase        0
SUS             0
Duration        0
Gender          0
ASR_Error       0
Intent_Error    0
dtype: int64
```

In [ ]:
```
# pearson: standard correlation coefficient
df.corr(method='pearson')['SUS'].sort_values()
```

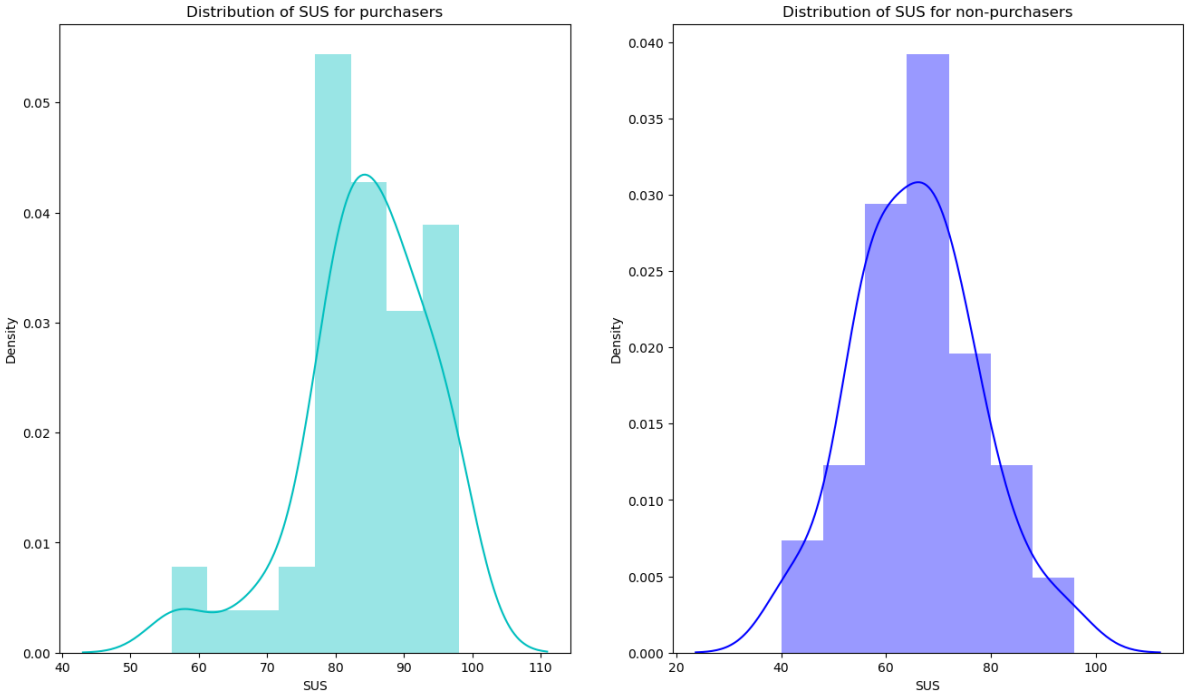
Out[ ]:
```
Intent_Error   -0.693675
ASR_Error      -0.662405
Duration       -0.006631
Gender          0.111523
Purchase        0.661931
SUS             1.000000
Name: SUS, dtype: float64
```

In [ ]:
```
# looking at SUS distribution
sns.histplot(data=df['SUS'])
plt.show()
```

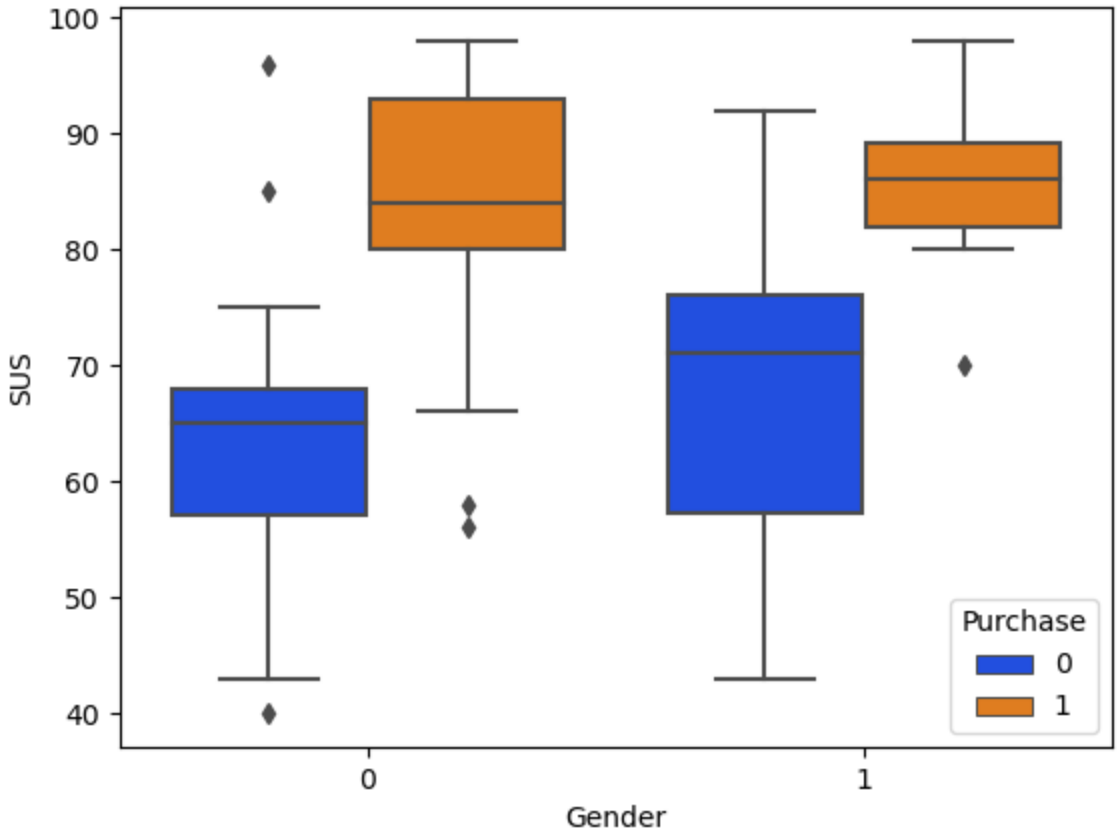


In [ ]:
```
# plot distribution of SUS for purchasers and non-purchasers

fig = plt.figure(figsize=(16,9))

ax1 = fig.add_subplot(121)
sns.distplot(df.loc[df['Purchase'] == 1]['SUS'], color='c')
ax1.set_title('Distribution of SUS for purchasers')

ax2 = fig.add_subplot(122)
sns.distplot(df.loc[df['Purchase'] == 0]['SUS'], color='b')
ax2.set_title('Distribution of SUS for non-purchasers')

plt.show()
```

```
/opt/anaconda3/envs/env/lib/python3.11/site-packages/seaborn/distributions.
py:2619: FutureWarning: `distplot` is a deprecated function and will be rem
oved in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-leve
l function for histograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/envs/env/lib/python3.11/site-packages/seaborn/distributions.
py:2619: FutureWarning: `distplot` is a deprecated function and will be rem
oved in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-leve
l function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]:
```
# compare SUS between genders of purchases
sns.boxplot(x="Gender", y="SUS", hue="Purchase", data=df, palette='bright')
plt.show() # 0-Female, 1-Male
```



In [ ]:
```
# use Ordinary Least Squares (OLS) Regression to find significant variables

import statsmodels.api as sm
```

```
y = df['SUS'] # dependent variable
x = df.drop(columns='SUS') # predictor variables

#add constant to predictor variables
x = sm.add_constant(x)

model = sm.OLS(y, x).fit()
print(model.summary());
```

OLS Regression Results

Dep. Variable:	SUS	R-squared:	0.593
Model:	OLS	Adj. R-squared:	0.571
Method:	Least Squares	F-statistic:	27.39
Date:	Sat, 11 Feb 2023	Prob (F-statistic):	5.25e-17
Time:	10:35:58	Log-Likelihood:	-362.39
No. Observations:	100	AIC:	736.8
Df Residuals:	94	BIC:	752.4
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	93.0282	5.541	16.788	0.000	82.026	104.031
Purchase	1.3412	3.676	0.365	0.716	-5.958	8.641
Duration	-0.0002	0.010	-0.025	0.980	-0.020	0.019
Gender	0.8367	1.971	0.425	0.672	-3.076	4.749
ASR_Error	-1.4254	0.401	-3.553	0.001	-2.222	-0.629
Intent_Error	-2.0092	0.439	-4.572	0.000	-2.882	-1.137

Omnibus:	6.969	Durbin-Watson:	2.023
Prob(Omnibus):	0.031	Jarque-Bera (JB):	8.115
Skew:	-0.378	Prob(JB):	0.0173
Kurtosis:	4.173	Cond. No.	1.27e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.27e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:
```
# Regression Analysis

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score,mean_squared_error
```

In [ ]:
```
x = df.drop(columns='SUS')
y = df['SUS']

x_train, x_test, y_train, y_test = train_test_split(x, y)
```

In [ ]:
```
# Lines, linear model prediction

lr = LinearRegression().fit(x_train,y_train)

y_train_pred = lr.predict(x_train)
y_test_pred = lr.predict(x_test)

print("The R squared score of linear regression model is: ", lr.score(x_test
```

```
The R squared score of linear regression model is:  0.37048508817295345
```

In [ ]:
```
# Quadratic

quad = PolynomialFeatures (degree = 2)
x_quad = quad.fit_transform(x)

X_train,X_test,Y_train,Y_test = train_test_split(x_quad,y, random_state = 0)

plr = LinearRegression().fit(X_train,Y_train)

Y_train_pred = plr.predict(X_train)
Y_test_pred = plr.predict(X_test)

print("The R square score of 2-order polynomial regression model is: ", plr.
```

```
The R square score of 2-order polynomial regression model is:  0.5220498503
429807
```