***Before Anything, Press Runtime and Press Run All***

**Week 10** Hypothesis Testing

```
#Import packages
import scipy.stats as stats
from scipy.stats import ttest_1samp
from scipy.stats import ttest_ind
from scipy.stats import ttest_rel
import pandas as pd
```

Load & Read Dataset

```
#load and read the dataset
df =pd.read_csv('exercise_dataset.csv')
df.head()
```

**Pearson's Correlation Coefficient**

Calculate Pearson correlation coefficient between 'duration' and 'Calories Burn' and confrim the bewlow hypothses:

Hypotheses

H0: There is no significant correlation between exercise time and calories burned.

H1: There is a significant correlation between exercise time and calories burned.

```
stats.pearsonr(df['Duration'], df['Calories Burn'])
```

There is no Correlation between the two as the result was 0, so I have confidence there is none.

**One-Sample T-test**

Assuming you want to test if the mean weight of participants is significantly different from the population mean weight (70 kg).

Hypothisis

H0: The mean weight of participants is equal to the population mean weight (70 kg).

H1: The mean weight of participants is significantly different from the population mean weight (70 kg).

```
stats.ttest_1samp(df['Actual Weight'], 70)
```

My Hypothsis Answer: H1 - The P Value is Less Than 0.05 due to the e-99 in the result

**Two-Sample T-test**

Assuming you want to test if there is a significant difference in exercise time between male and female participants.

Hypotheses

H0: There is no significant difference in exercise time between male and female participants.

H1: There is a significant difference in exercise time between male and female participants.

```
group1 = df[df['Gender']=='Male']
group2 = df[df['Gender']=='Female']

ttest_ind(group1['Duration'], group2['Duration'])
```

Your Hypothsis Answer: H0 as there is no significant difference in exercise time between Men and Women, P Value is greater than 0.05

**Paired T-test**

Assuming you want to test if there is a significant difference in weight before and after exercise.

- Hint (look at the dataset for the varaible that represnt the "after exercise")

Hypotheses

H0: There is no significant difference in weight before and after exercise.

H1: There is a significant difference in weight before and after exercise.

```
ttest_rel(df['Actual Weight'], df['Dream Weight'])
```

Your Hypothsis Answer: H0 - P Value is Greater than 0.05

**Week 8** Calculating Correlation Using Numpy

```
import numpy as np

x = np.arange(10, 20)
x

y= np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
y

c = np.corrcoef(x,y)
c
```

**Calculating Correlation Using Scipy**

```
import scipy.stats

x = np.arange(10, 20)
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
scipy.stats.pearsonr(x, y).correlation
```

```
scipy.stats.spearmanr(x,y)
```

```
scipy.stats.kendalltau(x,y)
```

**Calculating Correlation Using Pandas**

```
import pandas as pd
```

```
x = pd.Series(range(10, 20))
```

```
y = pd.Series([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
```

```
x.corr(y)
```

```
y.corr(x)
```

```
df= pd.read_csv('/content/sample_data/SF Salaries.csv')
```

```
df.head()
```

```
k= df.corr()
```

```
import seaborn as sns
```

```
sns.heatmap(k)
```

**Exploratory Data Analysis**

Importing the Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading in the File

```
file_path = "EcomCust.csv"
Customers = pd.read_csv(file_path)
```

Printing the Head of the Dataset

```
Customers.head()
```

Printing the Dataset Using Info

```
Customers.info()
```

Printing the Dataset Using Describe

```
Customers.describe()
```

Using Seaborn, Create JointPlot to Compare Columns

```
#Create JoinPLot
sns.set(style="whitegrid")
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=Customers)
#Show Plot
plt.show()
```

Use JointPlot to Create a 2D Hex Bin Plot to Compare 2 Columns

```
#Create 2D Hex
sns.set(style="whitegrid")  # Optional: Set the style for the plot
sns.jointplot(x='Time on App', y='Length of Membership', kind='hex', data=Customers)

#Show Plot
plt.show()
```

Use PairPlot to Recreate Plots

```
sns.pairplot(Customers)
```

Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

The Length of Membership seems to have the most tightly packed scatter in relation to the Yearly Amount Spent.

Create Linear Model Plot using Seaborn Lmplot to Compare Columns

```
sns.lmplot(x='Length of Membership', y='Yearly Amount Spent', data=Customers)
```

**Training & Testing Data**

Set Variable X for all Numerical, Y to a Single Column

```
#Select Numericals for X
X = Customers[["Avg. Session Length", "Time on App", "Time on Website", "Length of Membership"]]

#Select Target Value for Y
Y = Customers["Yearly Amount Spent"]
```

Use Training SKLearn to Split Data into Training & Testing

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=101)
```

**Training the Model**

Import LinearRegression

```
from sklearn.linear_model import LinearRegression
```

Create Instance of LinearRegression called LM (lm)

```
#Creating Instance of Regression
lm = LinearRegression()
```

Train & Fit LM (lm) on Training Data

```
lm.fit(X_train, Y_train)
```

Print Out the Coefficients of the Model

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

**Predicting Test Data**

```
from sklearn.metrics import mean_squared_error, r2_score
predictions = lm.predict(X_test)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")

# Calculate R-squared
r2 = r2_score(y_test, predictions)
print(f"R-squared: {r2}")
```

Create ScatterPlot of Real Test Values vs Predicted

```
plt.scatter(y_test, predictions)
plt.xlabel("True Values (y_test)")
plt.ylabel("Predicted Values")
plt.title("Scatter Plot of True vs. Predicted Values")
plt.show()
```

**Evaluating the Model**

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, predictions)
print(f"Mean Absolute Error: {mae}")

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")

# Calculate Root Mean Squared Error
rmse = mse**0.5
print(f"Root Mean Squared Error: {rmse}")
```

Plot Histogram of Residuals using Seaborn Distplot or PltHist

```
# Calculate residuals
residuals = y_test - predictions

# Plot histogram of residuals using Seaborn
sns.histplot(residuals, kde=True)
plt.xlabel("Yearly Amount Spent")
plt.ylabel("Density")
plt.show()
```

**Week 7** Graphing w/ Imports

Import Packages

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

titanic = sns.load_dataset('titanic')

titanic.head()
```

JointPlot:

```
sns.jointplot(x='fare', y='age', data=titanic, kind='scatter')
```

DistPlot:

```
sns.distplot(titanic['fare'],bins=30,kde=False,color='red')
```

StripPlot:

```
sns.stripplot(x='alive', y='age', data=titanic, hue = 'sex', jitter = True,)
```

BoxPlot:

```
sns.boxplot(x='alive', y='age', data=titanic, hue='sex', palette='coolwarm')
```

```
sns.boxplot(x='class', y='age', data=titanic, palette = 'rainbow')
```

```
sns.swarmplot(x='class',y='age',data=titanic,palette='Set2')
```

CountPlot:

```
sns.countplot(x='sex', data = titanic)
```

HeatMap:

```
sns.heatmap(titanic.corr(), cmap='coolwarm')
```

FacetGrid:

```
g = sns.FacetGrid(titanic, col='sex')
g.map(plt.hist, 'age')
```