code for condn prob

```
# Total number of records
total_records = 10
# Number of records where recreation is "golf"
number_golf_recreation = 4
prob_golf = number_golf_recreation / total_records
print("Unconditional probability of golf: ={}".format(prob_golf))

# Number of records with "medRisk" and number of records with both "medRisk" and "single"
number_med_risk_single = 2
number_med_risk = 3

# Calculate probabilities
prob_med_risk_single = number_med_risk_single / total_records
prob_med_risk = number_med_risk / total_records
conditional_probability = prob_med_risk_single / prob_med_risk

print("Conditional probability of single given medRisk: = {}".format(conditional_probability))
```

Code for linear regression

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Load the dataset
df = pd.read_csv('/path/to/bottle.csv')  # Change the path to the location of your CSV file

# Select only the 'Salnty' and 'T_degC' columns and rename them
df_binary = df[['Salnty', 'T_degC']]
df_binary.columns = ['Sal', 'Temp']

# Display the first 5 rows of the dataframe
print(df_binary.head())

# Plotting the scatter plot to check the relationship between Sal and Temp
sns.lmplot(x="Sal", y="Temp", data=df_binary, order=2, ci=None)
plt.show()

# Drop any rows with NaN values
df_binary.dropna(inplace=True)

# Separate the data into independent (X) and dependent (y) variables
# Convert each dataframe into a numpy array
X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Split the data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Create and fit the linear regression model
regr = LinearRegression()
regr.fit(X_train, y_train)

# Predict the test set results
y_pred = regr.predict(X_test)

# Print the R^2 score of the model on the test data
print("R^2 score:", regr.score(X_test, y_test))
```

```
# Calculate and print error metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
```

Naive bayes

```
 import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score

# Load the dataset
msg = pd.read_csv('document.csv', names=['message', 'label'])

# Display the total number of instances in the dataset
print("Total Instances of Dataset: ", msg.shape[0])

# Map label values to numerical values
msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})

# Separate the data into features (X) and labels (y)
X = msg.message
y = msg.labelnum

# Split the data into training and testing sets
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
# Vectorize the text data
count_v = CountVectorizer()
Xtrain_dm = count_v.fit_transform(Xtrain)
Xtest_dm = count_v.transform(Xtest)

# Create a DataFrame from the training data matrix (optional, for viewing the features)
df = pd.DataFrame(Xtrain_dm.toarray(), columns=count_v.get_feature_names_out())

# Train the Naive Bayes classifier
clf = MultinomialNB()
clf.fit(Xtrain_dm, ytrain)

# Predict the labels for the test set
pred = clf.predict(Xtest_dm)

# Print accuracy metrics
print('Accuracy Metrics:')
print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

BACKPROPOGATION

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score


# Load the dataset

msg = pd.read_csv('document.csv', names=['message', 'label'])


# Display the total number of instances in the dataset

print("Total Instances of Dataset: ", msg.shape[0])


# Map label values to numerical values

msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})


# Separate the data into features (X) and labels (y)

X = msg.message

y = msg.labelnum


# Split the data into training and testing sets

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
# Vectorize the text data

count_v = CountVectorizer()

Xtrain_dm = count_v.fit_transform(Xtrain)

Xtest_dm = count_v.transform(Xtest)



# Create a DataFrame from the training data matrix (optional, for viewing the features)

df = pd.DataFrame(Xtrain_dm.toarray(), columns=count_v.get_feature_names_out())



# Train the Naive Bayes classifier

clf = MultinomialNB()

clf.fit(Xtrain_dm, ytrain)



# Predict the labels for the test set

pred = clf.predict(Xtest_dm)



# Print accuracy metrics
```

```
print('Accuracy Metrics:')

print('Accuracy: ', accuracy_score(ytest, pred))

print('Recall: ', recall_score(ytest, pred))

print('Precision: ', precision_score(ytest, pred))

print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```