**Final Year B. Tech. (CSE) – I: 2022-23**

**5CS462: PE5 - Data Mining Lab**

**Assignment No. 2**

**PRN: 2019BTECS00077 Date:28 Aug 2022**

**Full name: Biradar Avinash Vishnu**

**Title: Correlation analysis using Chi square test,covariance,pearson coefficient and normalization of data**

**Objective: Design a GUI such that correlation between two attributes can be find out and normalization of the data**

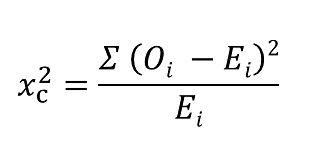
**Dataset Use: Iris, Brest Cancer, Employee Attrition**

**Introduction & Theory:**

**Chi Square Test**

The Chi-Square test is a statistical procedure for determining the difference between observed and expected [data.](https://www.simplilearn.com/what-is-data-article) This test can also be used to determine whether it correlates to the categorical variables in our data. It helps to find out whether a difference between two categorical variables is due to chance or a relationship between them.

**Formula For Chi-Square Test**



Where

c = Degrees of freedom

O = Observed Value

E = Expected Value

**Pearson Coefficient:**

The **Pearson correlation coefficient (r)** is the most common way of measuring a linear correlation. It is a number between –1 and 1 that measures the strength and direction of the relationship between two variables.

**Normalization Techniques:**

1. **Min -Max**
2. **Z-Score**
3. **Decimal Scaling**

**Implementation:**

if assignNo=="2":

        colums=df.columns

        st.header("Chi Sqaure Test")

        st.sidebar.subheader("Select Attributes For Chi square")

        att1= st.sidebar.selectbox("Select attribute 1",colums)

        att2= st.sidebar.selectbox("Select categarical attribute",colums)

        contigency= pd.crosstab(df[att1], df[att2],margins=True)

        st.subheader("Contingency/Observed Table")

        st.text(contigency)

        row\_len=len(df[att1].unique())

        col\_len=len(df[att2].unique())

        row\_sum = contigency.iloc[0:row\_len,col\_len].values

        exp = []

        for j in range(row\_len):

            for val in contigency.iloc[row\_len,0:col\_len].values:

                exp.append(val \* row\_sum[j] / contigency.loc['All', 'All'])

        obs=[]

        for j in range(row\_len):

            for val in contigency.iloc[j,0:col\_len].values:

                obs.append(val)

        #Expected Table

        expArr=np.array(exp).reshape(row\_len,col\_len)

        st.subheader("expected Table")

        st.write(expArr)

        #Degree of Freedom

        st.subheader('Degree of Freedom')

        st.write("no of Rows:",row\_len)

        st.write("no of Columns:",col\_len)

        degreeOfFreedom=(row\_len-1)\*(col\_len-1)

        st.write("(row-1)\*(col-1)=",degreeOfFreedom)

        #((obs[i] - exp[i])^2/exp[i])

        st.subheader('(Obs[i]-exp[i])^2/exp[i]')

        objmexp=[]

        chiSquareValue=0

        for i in range(len(obs)):

          chiSquareValue+=((obs[i] - exp[i])\*\*2/exp[i])

          objmexp.append((obs[i] - exp[i])\*\*2/exp[i])

        objmexp=np.array(objmexp).reshape(row\_len,col\_len)

        st.write(objmexp)

        #chi Square Value

        st.subheader("conclusion of Chi Sqaure Test")

        st.write("Chi sqaure Value:",chiSquareValue)

        criticalValue = scipy.stats.chi2.ppf(1-.001, df = degreeOfFreedom)

        st.write("Critical Value:",criticalValue)

        if(criticalValue > chiSquareValue):

            st.write("chiSquare Value is less than critical Value so,They are independent")

        else:

            st.write("chiSquare Value is greater than critical Value so,They are Correlated")

        # covariance

        st.sidebar.subheader("Select Attr For Covaraince & Pearson")

        attr1= st.sidebar.selectbox("Select attr 1",colums)

        attr2= st.sidebar.selectbox("Select attr 2",colums)

        st.header(" covariance")

        data1=df[attr1].to\_list()

        data2=df[attr2].to\_list()

        xm =helper.mean(data1)

        ym=helper.mean(data2)

        n=len(data1)

        covariance=0.0

        for i in range(n):

            covariance += (data1[i]-xm)\*(data2[i]-ym)/(n-1)

        st.write("The Covariance is :",covariance)

        # Pearson coefficient

        st.header("Pearson coefficient")

        stdD1=helper.stddeviation(data1)

        stdD2=helper.stddeviation(data2)

        pearson=(covariance/(stdD1\*stdD2))

        st.write("The Pearson coefficient is :",pearson)

        #conclusion of perason test

        res=""

        if(pearson>0):

          res="Positively Correlated"

        elif(pearson<0):

          res="Negatively Correlated"

        else:

          res="Independent"

        st.write("conclusion:",res)

        #Normalization

        st.header("Normalization")

        st.subheader("(Decimal Scaling,Min Max,Z-Score)")

        st.text("")

        st.sidebar.subheader("Select Attr For Normalization")

        att= st.sidebar.selectbox("Select attr",colums)

        #decimal Scaling

        st.header("1.Decimal Scaling")

        data=df[attr1].to\_list()

        n=len(data)

        denom=pow(10,len(str(max(data))))

        decimal\_scaling=[]

        for val in data:

            decimal\_scaling.append(val/denom)

        decimal\_scaling.sort()

        st.text(decimal\_scaling)

        #scatter plot of decimal scaling

        plt.locator\_params(nbins = 10)

        plt.scatter(decimal\_scaling,decimal\_scaling, c ="green", s=5)

        plt.xlabel(att)

        plt.ylabel(att)

        # plt.rcParams['figure.figsize'] = [8, 4]

        st.write("Scatter Plot")

        st.pyplot(plt)

        plt.clf()

        #Min-Max Scaling

        st.header("2.Min-Max Normalization")

        xmin=min(data)

        xmax=max(data)

        lmin=0 #local min

        lmax=1 #local max

        minMax=[]

        if xmin==xmax:

            st.write("denominator became zero because min and max are same")

        else:

            for val in data:

                minMax.append((val-xmin)/(xmax-xmin)\*(lmax-lmin)+lmin)

        st.text(minMax)

        #scatter plot of decimal scaling

        plt.locator\_params(nbins = 10)

        plt.scatter(minMax,minMax, c ="green", s=5)

        plt.xlabel(att)

        plt.ylabel(att)

        # plt.rcParams['figure.figsize'] = [8, 4]

        st.write("Scatter Plot")

        st.pyplot(plt)

        plt.clf()

        #Z-Score Scaling

        st.header("3.Z-Score Normalization")

        mean=helper.mean(data)

        stdD=helper.stddeviation(data)

        z\_score=[]

        for val in data:

            z\_score.append((val-mean)/stdD)

        st.text(z\_score)

        #scatter plot of decimal scaling

        plt.locator\_params(nbins = 10)

        plt.scatter(z\_score,z\_score, c ="green", s=5)

        plt.xlabel(att)

        plt.ylabel(att)

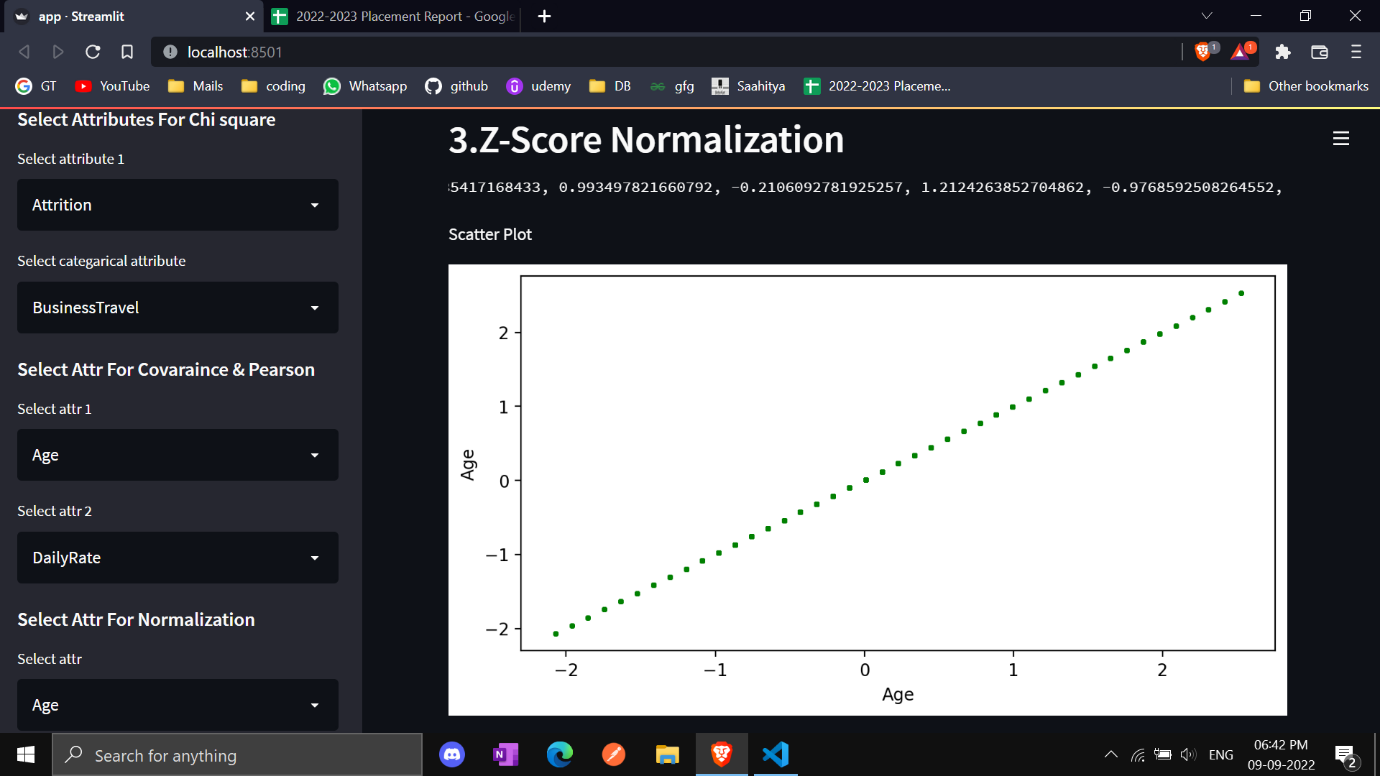
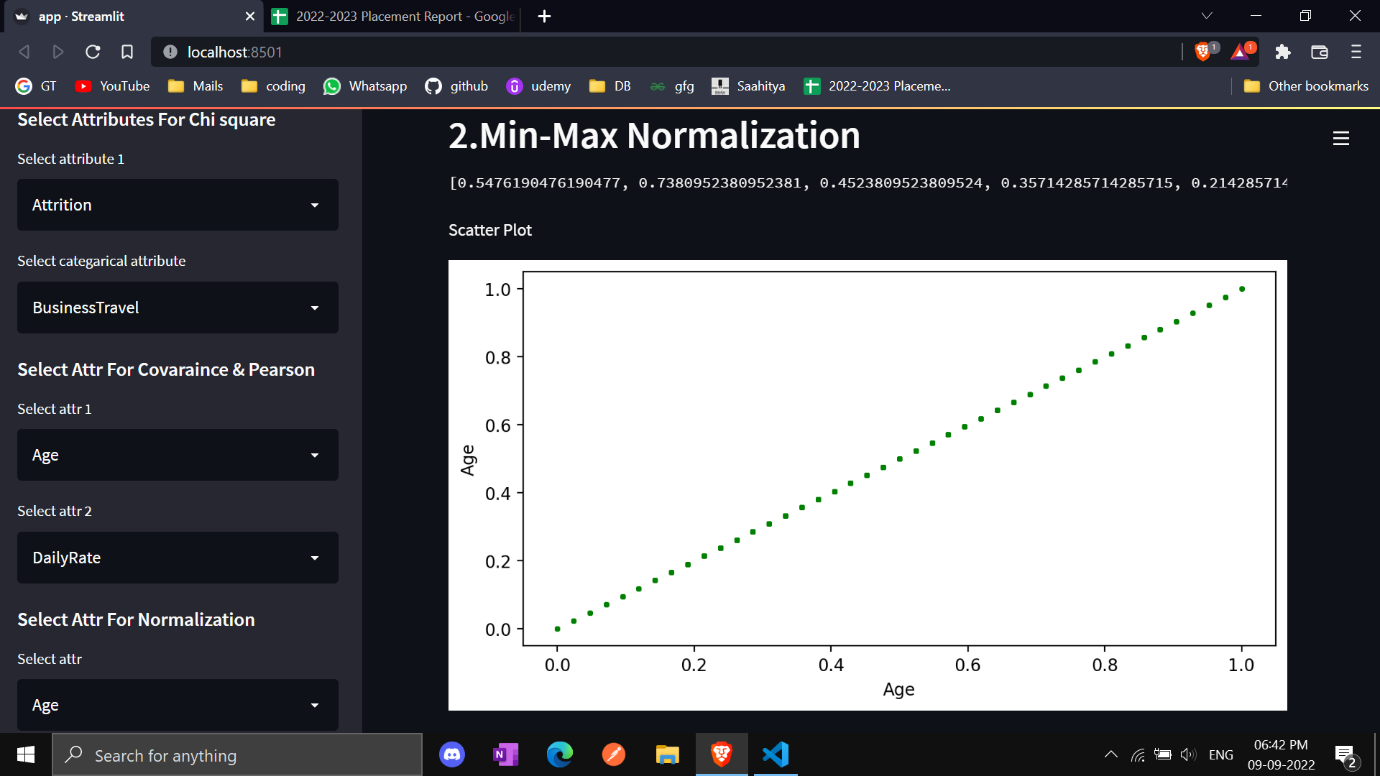
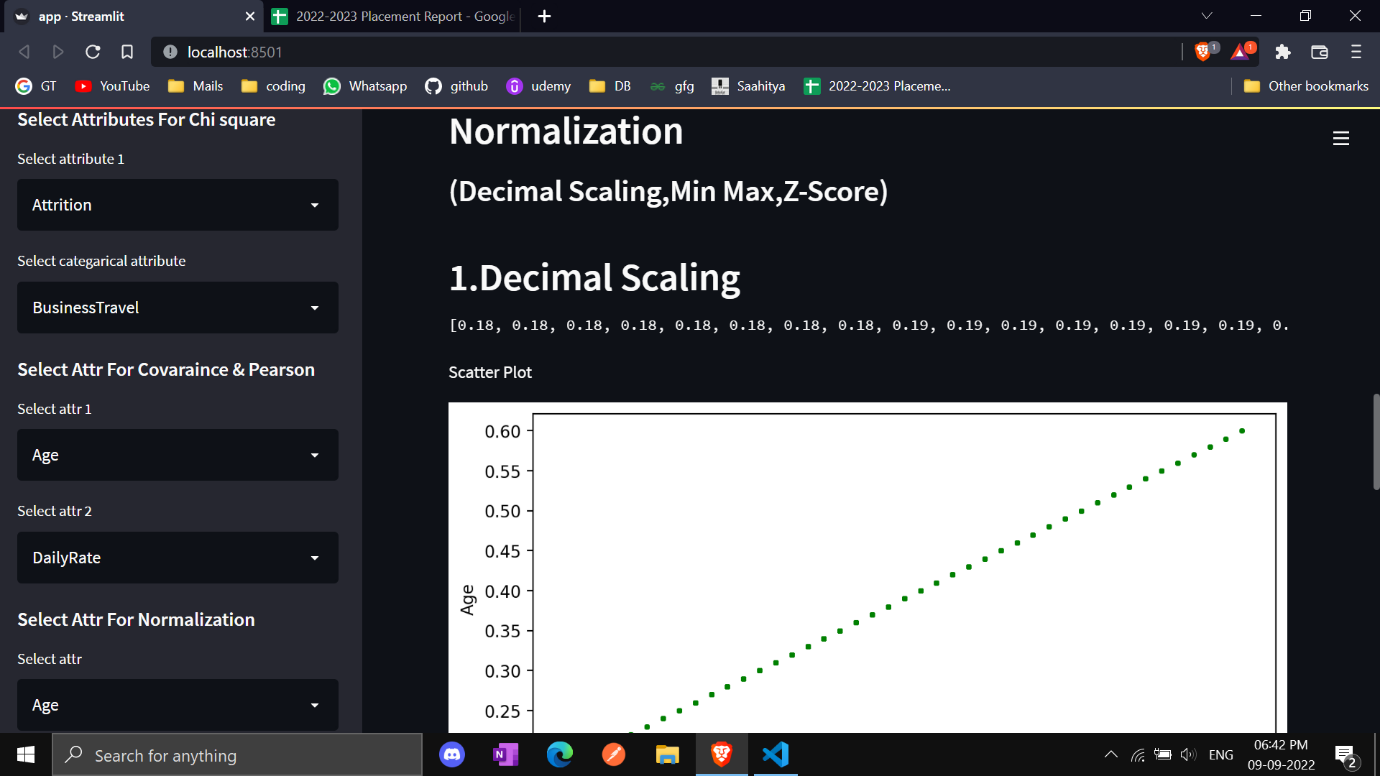
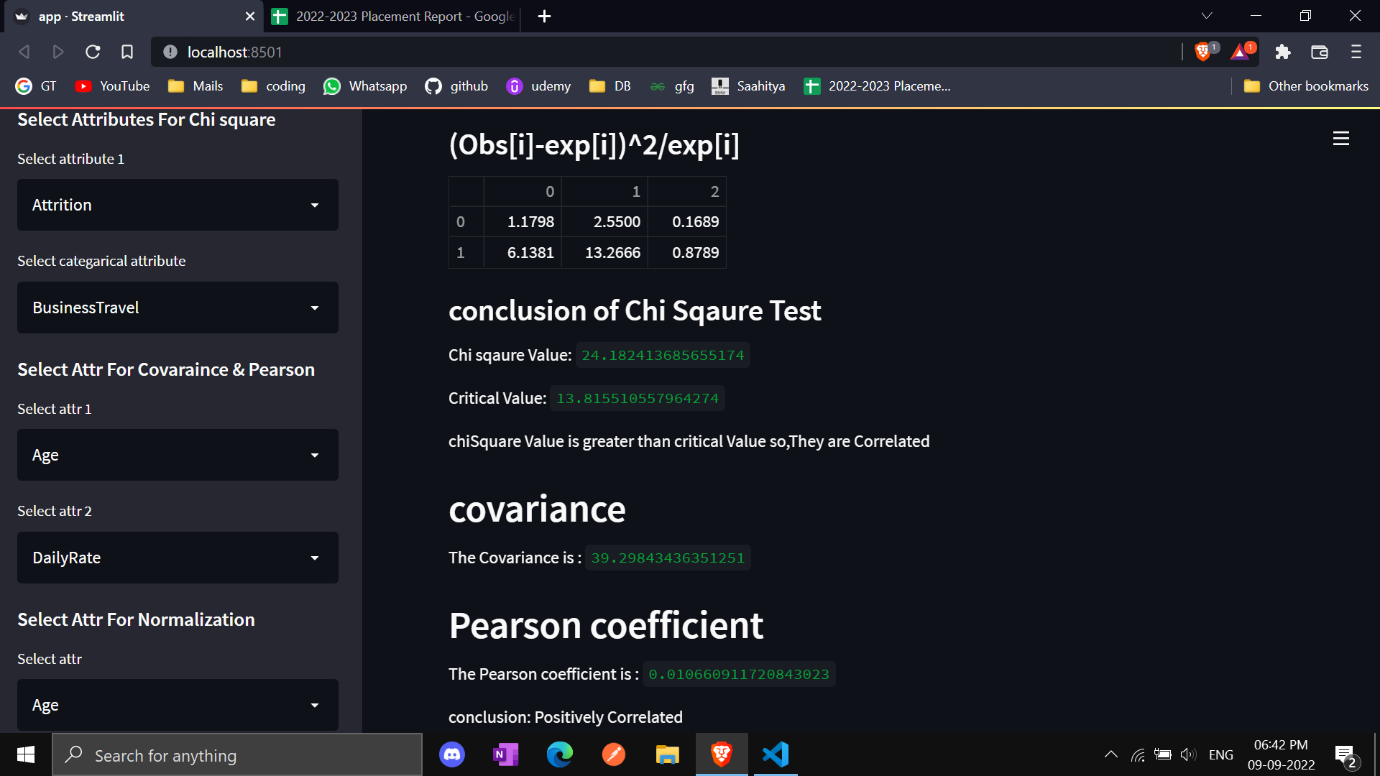
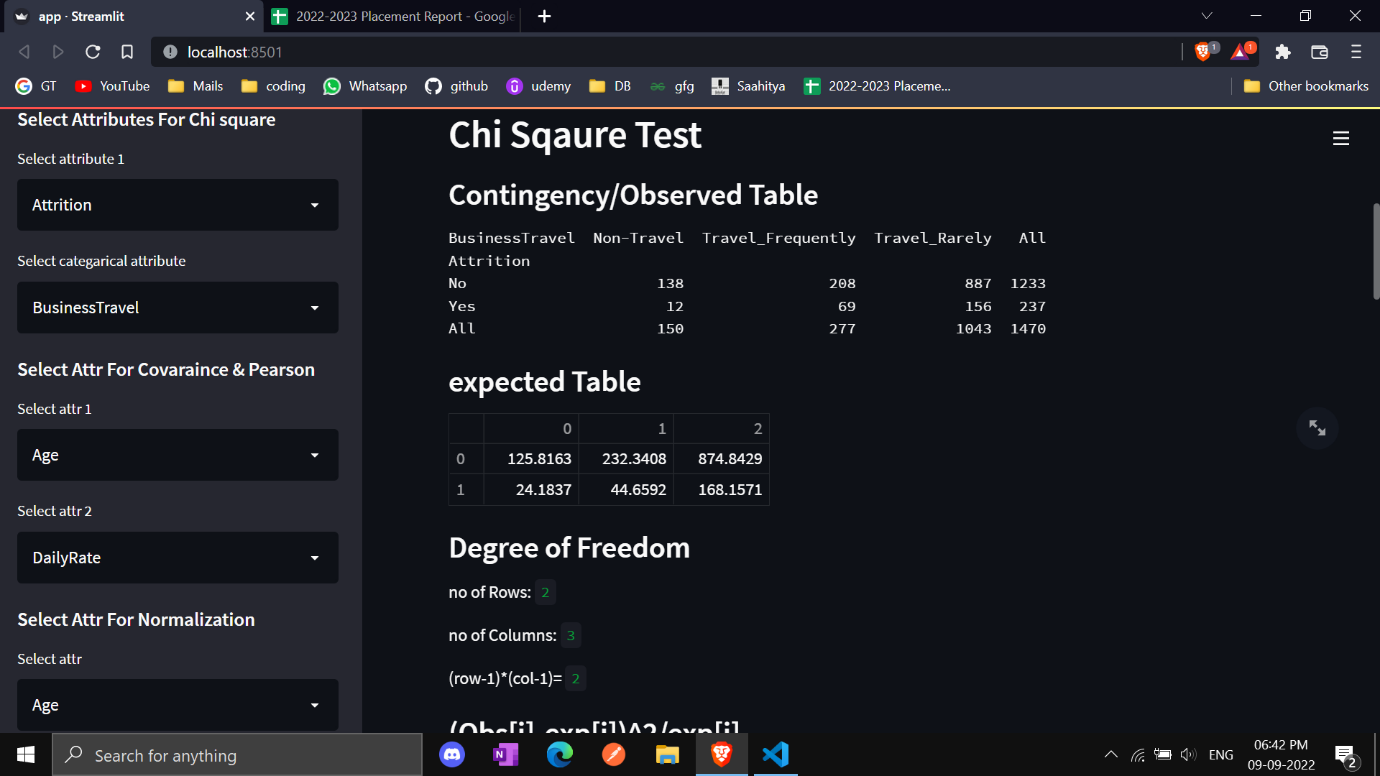
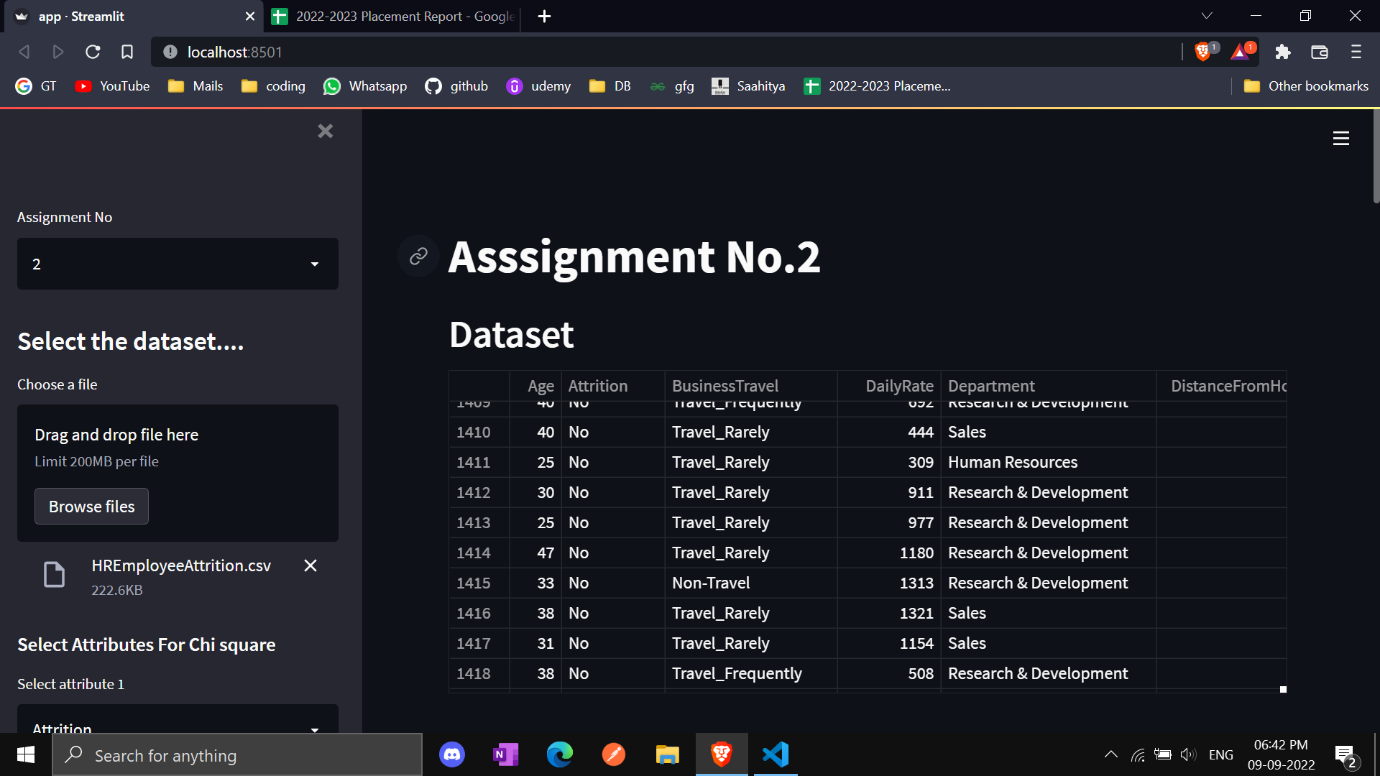
        # plt.rcParams['figure.figsize'] = [8, 4]

        st.write("Scatter Plot")

        st.pyplot(plt)

        plt.clf(

**ScreenShots of Output:**

****