Lab 6 README

The objective of this lab is to practice data science analysis for different scenarios using python such as:

-loans

-salary

-data relations on Caltrans parameters

-CoVid-19 trend prediction

The required libraries for this lab:

pandas, seaborn, matplotlib, numpy, scipy, sklearn, collections.abc, statistics, random, tensorflow, keras, MinMaxscaler

1) Risk management:
   The python program will analyze and identify the risk factor of which parameters will cause a loan to default and report back to a bank which set of parameters will be low, medium, high risk. The python code will run and an extra column in the excel sheet will tell the bank the risk of a person's loan.
   What is considered low risk, medium risk, high risk?
   We can use the p value to show correlation. a pvalue of above 0.05 would be consider two datasets to have correlation. Let's say we are comparing BAD with LOAN. it has a p value fo 0.07 which menas there is correlation. We can see from the graph that as the amount/loan increases, the number of people that default will becomes smaller meaning that it is lower risk. Then let's say we are comparing BAD vs the number of credit lines. We can see that as the number of active lines increase there are more people that will not pay off the loan, indicated that people tha have more credit lines open are at a higher risk of defaulting. Finally, lets say we are comparing BAD to year on the job, we see that there isnt much correlation between the two datasets. We can see that the higher number of years on the job does not show an inherent risk in if the person will default or pay off the loan.

967	5959	0	89900	48811	88934	DebtCon	Other	15	0	0	219.601	0	16	34.5715191	0.02060694	Low Risk
968	1	1	1300	70053	68400	HomeImp	Other	7	0	2	121.8333	0	14		0.236060271	Medium Risk

Figure 1: low risk vs medium risk

Figure 1 shows an example of what to take note of how the code will determine if a loan is high or low risk. The code is looking the number of credit lines, loan amount the age of the oldest credit lines. Even though loan#5959 has a high loan amount, its lower risk given the age of the oldest credit line.

480	1	7000	19894	32500	DebtCon	Other	4	0	0	134.7	1	16		0.241635405	Medium Risk
169	0	5000	39000	50340	HomeImp	Office	10			179.8333		19		0.241667371	High Risk

Figure 2: Medium risk vs High risk

Figure 2 shows an example of medium vs high risk. We can tell loan# 169 is considered high risk due to the largest amount of credit lines given the loan amount.

5953	0	88800	53307	94058	DebtCon	Other	16	0	0	218.305	0	15	34.2424647	0.021691884	Low Risk
163	1	5000	6000	38575	HomeImp	Self	7	0	1	335.0333	3	42		0.267559595	High Risk

Figure 3:Low risk vs High risk

Figure 3 shows an example of low risk vs high risk. When a set of data is fully provided, an average of the set of parameters will be used to calculate the risk factor of a loan. Given loan#2 is missing a debt to income ratio, an average has been used.

2) Data Relationships for Caltrans
We are finding a relationship between the distance traveled, delays, incidents, and locations of Caltrans service.

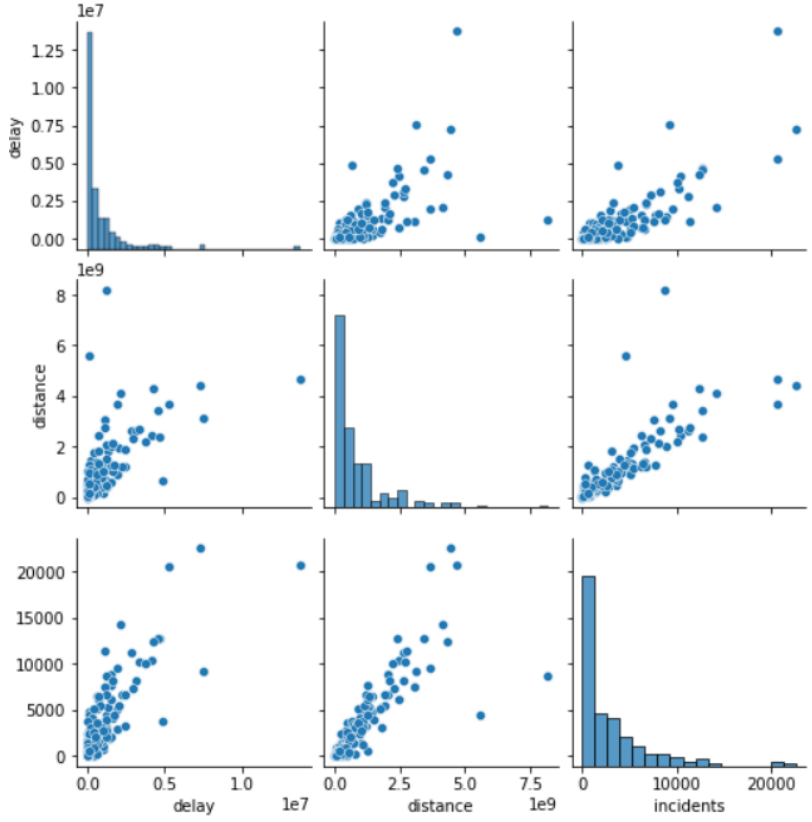

Figure 4: Relationships of caltrans parameters

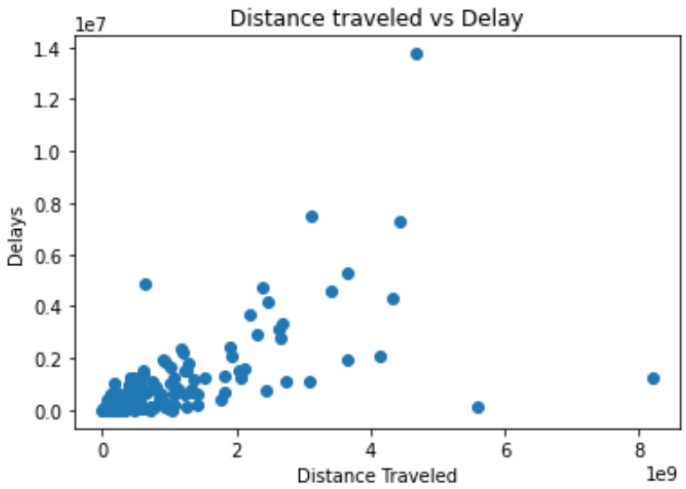

Figure 5: Distance traveled vs delays

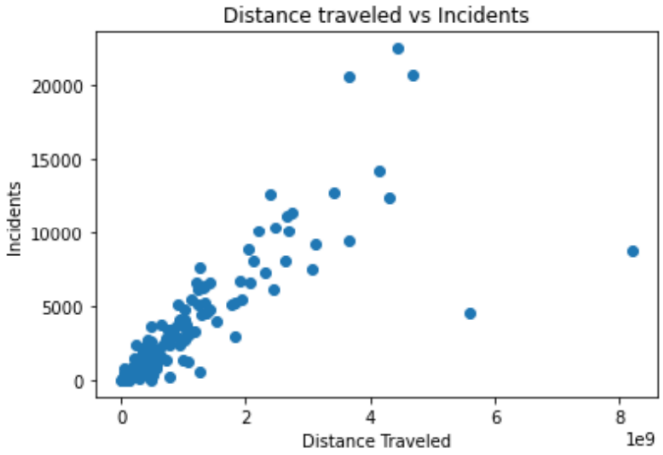

Figure 6: Distance traveled vs incidents

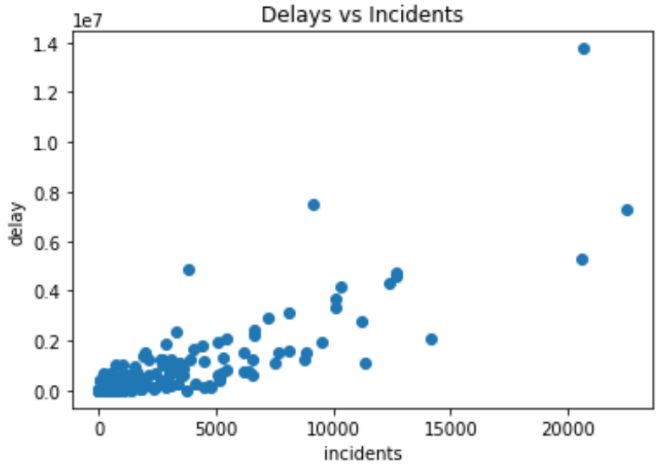

Figure 7:Delays vs incidents

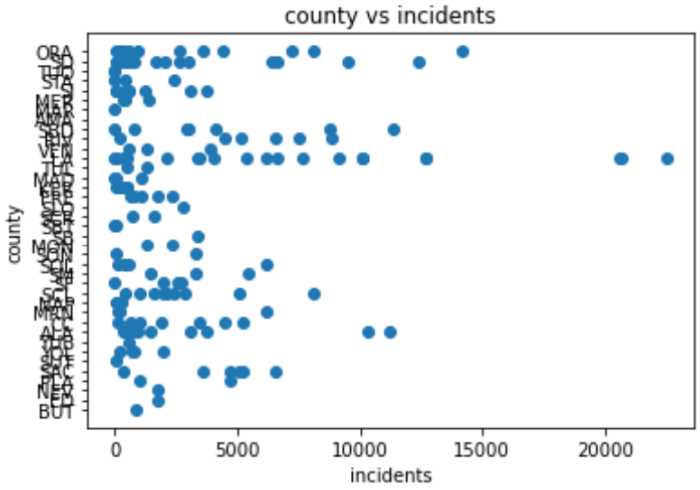

Figure 8: County vs incidents

From the data, we can observe that many of the variables have linear relationship with one another. We wanted to use an independent variable in counties to observe and additional element in Caltrans. By comparing counties to the number of incidents, we can conclude that distance may not always equate to more accidents and delays, it could be a short route in a more dangerous area as a catalyst for more delays and incidents

3) Normal distributions on San Francisco Salary
   We wanted to see how many people made a what salary in San Francisco to determine the median and mean of salaries in San Francisco. We only looked at base pay as overtime is inherent to additional hours. A person with enough overtime hours can earn more money than a more educated person's higher base pay.

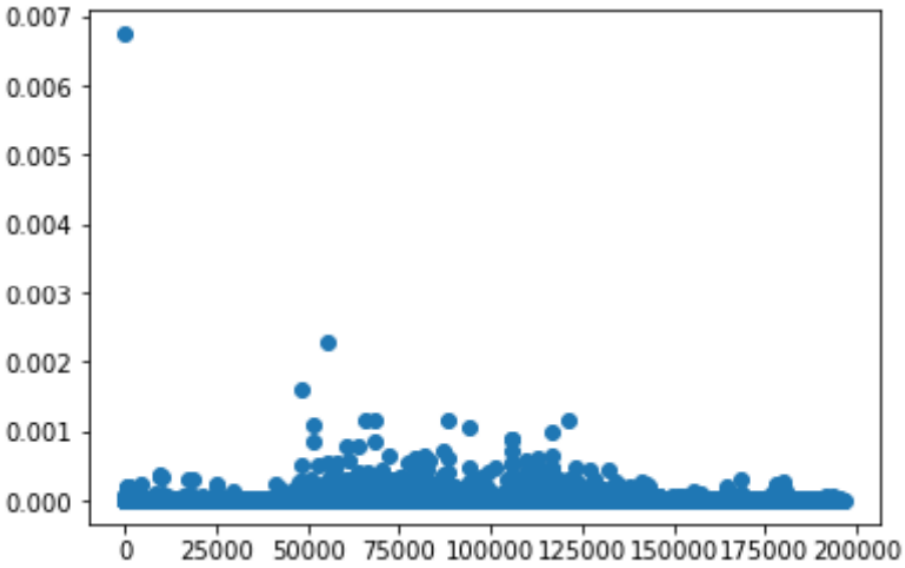

Figure 9: Salary graph

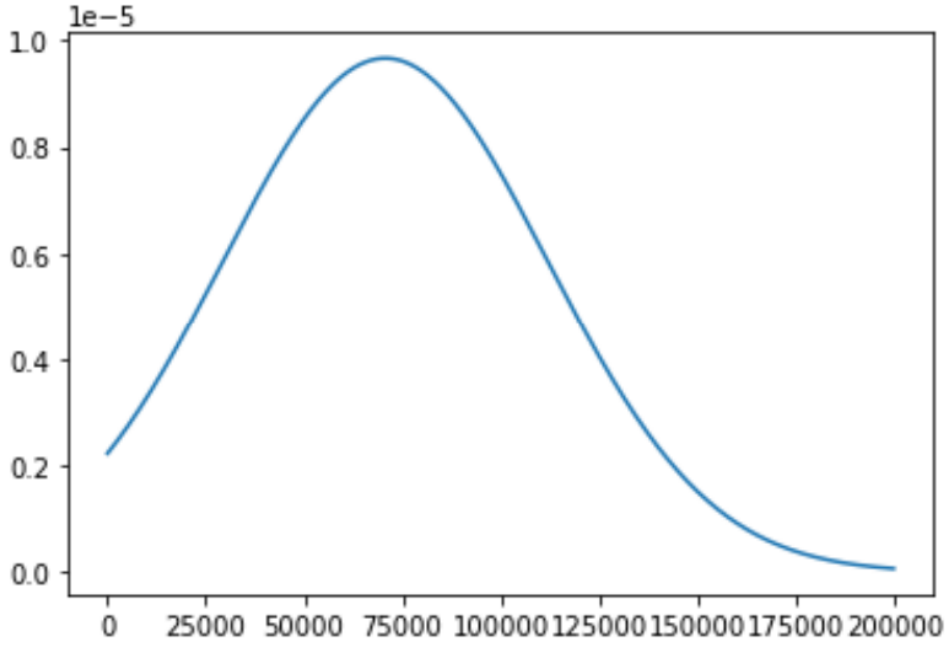

Figure 10: Normal Distribution

```
median: 69299.04
mean: 71323.83957370375
standard deviation: 42408.55005222362
number of people in the 1st, 2nd, and 3rd standard deviation 31092
number of people in the 1st, 2nd, and 3rd standard deviation 83546
number of people in the 1st, 2nd, and 3rd standard deviation 119128
```

Figure 11: Normalized numbrs

Outliers outside 3 standard deviations are removed, we were able to identify the average and median income in San Francisco along woth how many people are in each standard deviation salary bracket are.

4) Covid-19 data prediction
   By reading three months of previous Covid-19 data, we built a model that was able to predict the trend for the next three months of covid-19 data. By increasing the number of epochs or training intervals, we are able to produce a more accurate model.

Figure 12: Covid-19 prediction

Documentation:

1) Risk factor

- We first load the csv into the code ad read it with pandas, then we changed the labels to values under the BAD category, 0=paid off loan, 1=defaulted

```
df= pd.read_csv('hmeq.csv')
df.BAD.mask(df.BAD == 0,"Cleared")
df.BAD.mask(df.BAD == 1,"Defaulted")
df.BAD = le.fit_transform(df.BAD)
```

- We would compare bad with other data to find a correlation by calculating the p value, then we create a linear regression for all data with a correlation

```
pear_columns = []
for dataset in pear.iteritems():
    # print(test["BAD"])
    # print(test[dataset[0]])
    print("testing BAD vs ",dataset[0])
    res = (linregress(test["BAD"],test[dataset[0]])) ## All pvalues are great than 0.05
    print(res)
    if(res.pvalue > 0.05):
        pear_columns.append(dataset[0])
# print(np.corrcoef(test["LOAN"].dropna(),test["MORTDUE"]))
```

- Add a column into the excel sheet called predicted_BAD
- Risk is determined mean and standard deviation

```
df["Predicted_BAD"] = pred



pd.crosstab(df.Predicted_BAD,df.BAD,)
print(chi2_contingency(pd.crosstab(df.Predicted_BAD,df.BAD,)
))
plt.scatter(df.Predicted_BAD,df.BAD) ## Two groups, the higher the
plt.show()



mean = df.Predicted_BAD[df.BAD == 1].mean()
std = df.Predicted_BAD[df.BAD == 1].std()
HighRiskBound = mean + std
LowRiskBound = mean - std



highRiskMask = df.Predicted_BAD >= HighRiskBound
lowRiskMask = df.Predicted_BAD <= LowRiskBound
mediumRiskMask= ~(lowRiskMask | highRiskMask)
df["RiskLevel"]=np.zeros(len(df["BAD"]))
df["RiskLevel"] = df.RiskLevel.mask(highRiskMask,"High Risk")
df["RiskLevel"] = df.RiskLevel.mask(lowRiskMask,"Low Risk")
df["RiskLevel"] = df.RiskLevel.mask(mediumRiskMask,"Medium Risk")
```

2) Data relationships

-read the csv and determine independent and dependant variables that are worth comparing

```
dat = pd.read_csv("caltrains.csv").replace(' -    ',np.nan)
print('Data:(Rows, Columns)')
print(dat.shape)
dat.head(5)


# df = dat[['route','suffix','county','rank','delay','dista
dat = dat[['delay','distance','incidents','county']]
dat["incidents"]=pd.to_numeric(dat.incidents)
dat["delay"]=pd.to_numeric(dat.delay)
sns.pairplot(dat, kind="scatter")
plt.show()
```

-plot the graphs and look for a linear relationship

```
print(chi2_contingency(pd.crosstab(dat.distance,dat.incidents)))
plt.scatter(dat.distance,dat.delay)
plt.title('Distance traveled vs Incidents')
plt.xlabel('Distance Traveled')
plt.ylabel('incidents')
plt.show()
```

3) Normal Distribution

- Read the csv and determine which variable to compute

```
data = pd.read_csv("Salaries.csv")
print('Data:(Rows, Columns)')
print(data.shape)
data.head(5)

basepay=data[['BasePay']].to_numpy()
```

- Compute median, mean, and standard deviation

```
basepay=data.BasePay[~data.BasePay.isna() & ~(data.BasePay.apply(lambda x: isinstance(x,str)))]
print("median:",np.median(basepay))
print("mean:",np.mean(basepay))
print("standard deviation:", stats.stdev(basepay))
```

- Clean the data

```
def removeOutliers(series:pd.Series):
    med = np.median(series)
    std = np.std(series)
    return series[np.abs(med-series)<(3*std)]
```

- Plot the normal distribution and print the cleaned mean. Median, and standard deviation

```
def plotFreq(series:pd.Series):
    counts=(series.value_counts(normalize=True))
    plt.scatter(counts.index.values,counts.values)
```

```
plotFreq(removeOutliers(basepay))
plt.show()
x_axis = np.arange(0,200000,100)
mean = stats.mean(removeOutliers(basepay))
sd = stats.stdev(removeOutliers(basepay))
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()

sigmas=np.array([1,2,3])*np.std(basepay)
for sigma in sigmas:
    print("number of people in the 1st, 2nd, and 3rd standard deviation",(removeOutliers(basepay)<sigma).sum())
```

4) Covid-19 prediction

- Read the csv and determine how much data will be used train, plot the training data

```
#First we read the csv to plot the damily confirmed cases
df_3_months = pd.read_csv("california-history.csv",skiprows=range(1,120),nrows=92)

#Selecting only the columns for date and US case values
df_3_months = df_3_months[['date', 'positiveIncrease']]
print('\n')
print(df_3_months)

#Plotting our first 3 months of data from 3/01/2020 to 5/30/2020
plt.plot(df_3_months['positiveIncrease'])
plt.xlabel('Days')
plt.ylabel('Confirmed Cases')
plt.title('Confirmed Covid-19 Cases in the US (March to May)')
plt.xticks(rotation=70)
plt.show()
```

- Training includes changing the training data into arrays, converting it to values between 0 and 1 and using a optimizer like adam.

```
train_data = scaled_data[0:training_data_len, :]
x_train = []
y_train = []
for i in range(90, len(train_data)):
    x_train.append(train_data[i-90:i,0])
    y_train.append(train_data[i,0])

#Converting training data sets to numpy arrays
x_train = np.array(x_train)
y_train = np.array(y_train)

#Reshaping training set
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))

#Creating LSTM model
model=Sequential()
model.add(LSTM(50,return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(50,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

#Compiling model
model.compile(optimizer='adam', loss='mean_squared_error')
```