DAFTAR PUSTAKA

[1] Limantoro, W. S., Fatichah, C., & Yuhana, L. (2016). Rancang Bangun Aplikasi Pendeteksi Suara Tangisan Bayi. *JURNAL TEKNIK ITS*.

[2] I. S. DEWI, A. ZULKARNAIN, A. A. L. (2018). Identifikasi Suara Tangisan Bayi menggunakan Metode LPC dan Euclidean Distance. *Teknologi Nasional Bandung, Langlangbuana*.

[3] Manfredi, C., Bandini, A., Melino, D., Viellevoye, R., Kalenga, M., & Orlandi, S. (2018). Biomedical Signal Processing and Control Automated detection and classification of basic shapes of newborn cry melody. *Biomedical Signal Processing and Control*, *45*, 174–181. https://doi.org/10.1016/j.bspc.2018.05.033

[4] Dunstan, P. (n.d.). *open up and discover your baby's language*.

[5] Singh, N. (2012). MFCC and Prosodic Feature Extraction Techniques : A Comparative Study MFCC and Prosodic Feature Extraction Techniques : A Comparative Study. *International Journal of Computer Applications*, (February 2016). https://doi.org/10.5120/8529-2061

[6] Jaya, M. T. S., Puspitaningrum, D., & Susilo, B. (2016). PENERAPAN SPEECH RECOGNITION PADA PERMAINAN TEKA-TEKI SILANG MENGGUNAKAN METODE HIDDEN MARKOV MODEL (HMM) BERBASIS DESKTOP. *Jurnal Rekursif*.

[7] Rohman, A. (2015). Model Algoritma K-Nearest Neighbor (K-Nn) Untuk Prediksi Kelulusan Mahasiswa. *Neo Teknika*, *1*(1). http://doi.org/10.1017/CBO9781107415324.004

[8] Sistem, J., Bisnis, I., & Kupang, S. U. (2015). Implementasi Algoritma K-Nearest Neighbor Sebagai Pendukung Keputusan Klasifikasi Penerima Beasiswa PPA dan BBM. *Jurnal Sistem Informasi Bisnis*.

[9] S. M. E. Hutahaean. (2017). Bayi Sensitivitas Ibu Usia Remaja Yang Memiliki 0-3 Tahun. *Universitas Sanata Dharma*.

[10] Weninger, F. (2016). open-Source Media Interpretation by Large feature-space Extraction, (November).

[11] Cohen, R. (2012). Infant Cry Analysis and Detection. *Convention of Electrical and Electronics Engineers in Israel*.

[12] Z, A. A., Hidayatno, A., Widyanto, M., & Saksono, T. (2008). Dengan Pengendali Jarak Jauh. *Jurnal Teknik Elektro*.

[13] Kheddache, Y., & Tadj, C. (2019). Biomedical Signal Processing and Control Identification of diseases in newborns using advanced acoustic features of cry signals. *Biomedical Signal Processing and Control*, *50*, 35–44. https://doi.org/10.1016/j.bspc.2019.01.010

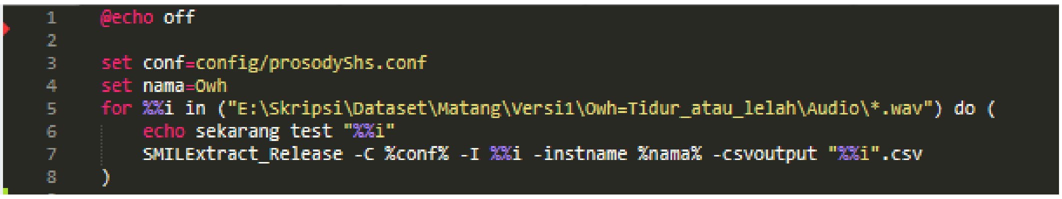
[14] Shriberg, E., & Ferrer, L. (2005). Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, *46*, 455–472. https://doi.org/10.1016/j.specom.2005.02.018

[15] Huang, Z., Chen, L., & Harper, M. (2006). An Open Source Prosodic Feature Extraction Tool. *School of Electrical and Computer Engineering*.

[16] Beasiswa, P., Misi, B., & Polbeng, D. I. (2016). *K-MEANS UNTUK MENENTUKAN CALON*. *1*(1).

LAMPIRAN

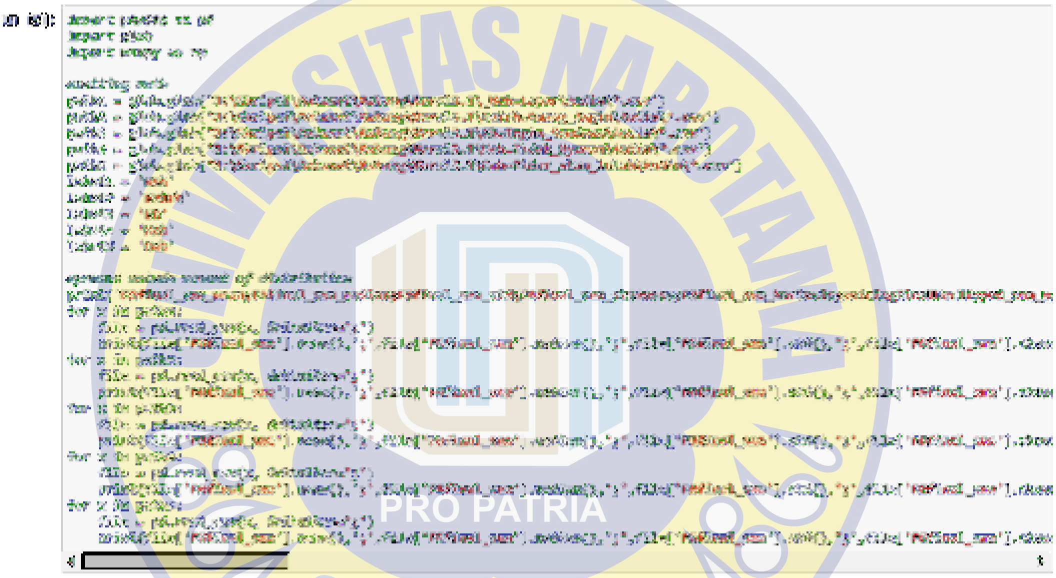
Lampiran 1: *Source Code extraction features*



```
@echo off

set conf=config/prosodyShs.conf
set nama=Owh
for %%i in ("E:\Skripsi\Dataset\Matang\Versi1\Owh=Tidur_atau_lelah\Audio\*.wav") do (
    echo sekarang test "%%i"
    SMILExtract_Release -C %conf% -I %%i -instname %nama% -csvoutput "%%i".csv
)
```

Lampiran 2 : *Source Code Moments of Distribution*

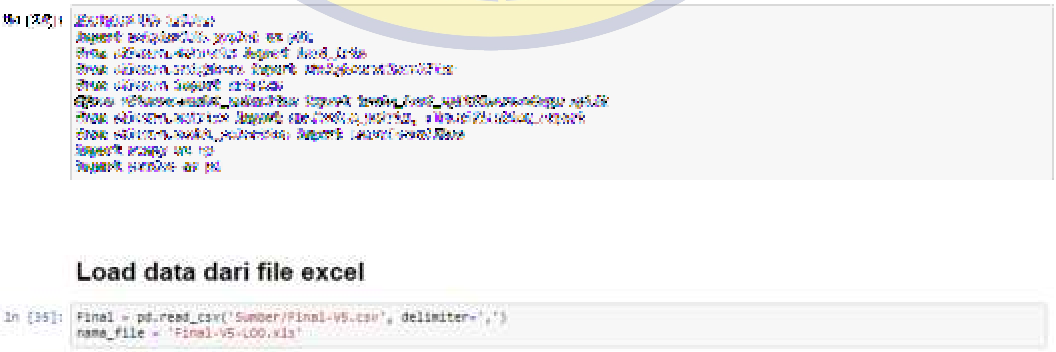


```
In [8]: import pandas as pd
        import glob
        import numpy as np

        #setting path
        path1 = glob.glob("E:\Skripsi\Dataset\Matang\Versi1\[illegible]\*.csv")
        path2 = glob.glob("E:\Skripsi\Dataset\Matang\Versi1\[illegible]\*.csv")
        path3 = glob.glob("E:\Skripsi\Dataset\Matang\Versi1\[illegible]\*.csv")
        path4 = glob.glob("E:\Skripsi\Dataset\Matang\Versi1\[illegible]\*.csv")
        path5 = glob.glob("E:\Skripsi\Dataset\Matang\Versi1\Owh=Tidur_atau_lelah\Audio\*.csv")
        label1 = 'Neh'
        label2 = 'Eairh'
        label3 = 'Eh'
        label4 = 'Heh'
        label5 = 'Owh'

        #proses untuk moment of distribution
        print('F0final_sma_amean;F0final_sma_median;F0final_sma_stddev;F0final_sma_skewness;F0final_sma_kurtosis;[illegible]
        for x in path1:
            file = pd.read_csv(x, delimiter=';')
            print(file['F0final_sma'].mean(),';',file['F0final_sma'].median(),';',file['F0final_sma'].std(),';',file['F0final_sma'].skew
        for x in path2:
            file = pd.read_csv(x, delimiter=';')
            print(file['F0final_sma'].mean(),';',file['F0final_sma'].median(),';',file['F0final_sma'].std(),';',file['F0final_sma'].skew
        for x in path3:
            file = pd.read_csv(x, delimiter=';')
            print(file['F0final_sma'].mean(),';',file['F0final_sma'].median(),';',file['F0final_sma'].std(),';',file['F0final_sma'].skew
        for x in path4:
            file = pd.read_csv(x, delimiter=';')
            print(file['F0final_sma'].mean(),';',file['F0final_sma'].median(),';',file['F0final_sma'].std(),';',file['F0final_sma'].skew
        for x in path5:
            file = pd.read_csv(x, delimiter=';')
            print(file['F0final_sma'].mean(),';',file['F0final_sma'].median(),';',file['F0final_sma'].std(),';',file['F0final_sma'].skew
```

Lampiran 2 : *Source Code K-Nearest Neighbours* dengan data sampling LOO

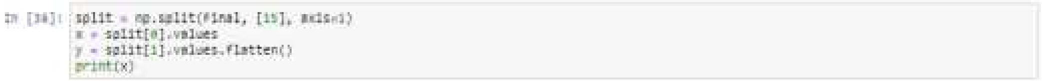


```
In [38]: #Import the library
         import matplotlib.pyplot as plt
         from sklearn.datasets import load_iris
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn import metrics
         from sklearn.model_selection import train_test_split, LeaveOneOut
         from sklearn.metrics import accuracy_score, classification_report
         from sklearn.model_selection import cross_val_score
         import numpy as np
         import pandas as pd
```

Load data dari file excel

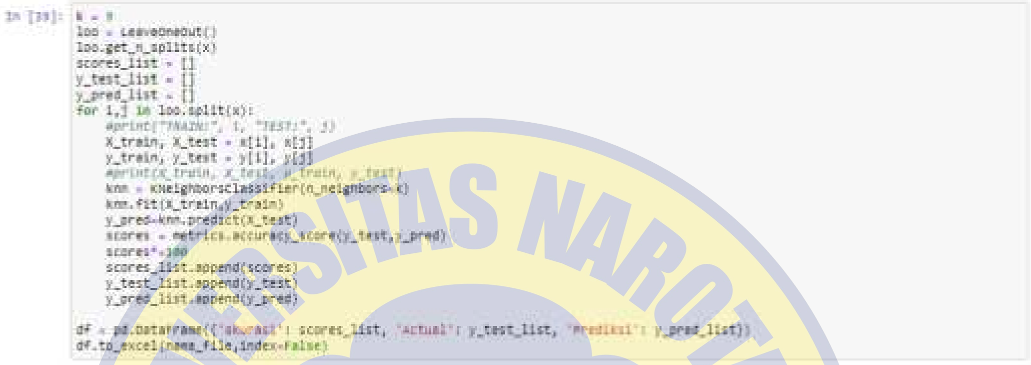
```
In [35]: Final = pd.read_csv('Sumber/Final-V5.csv', delimiter=',')
         nama_file = 'Final-V5-LOO.xls'
```

Bagi data menjadi target dan data dari fitur ¶



```
In [38]: split = np.split(final, [15], axis=1)
         x = split[0].values
         y = split[1].values.flatten()
         print(x)
```

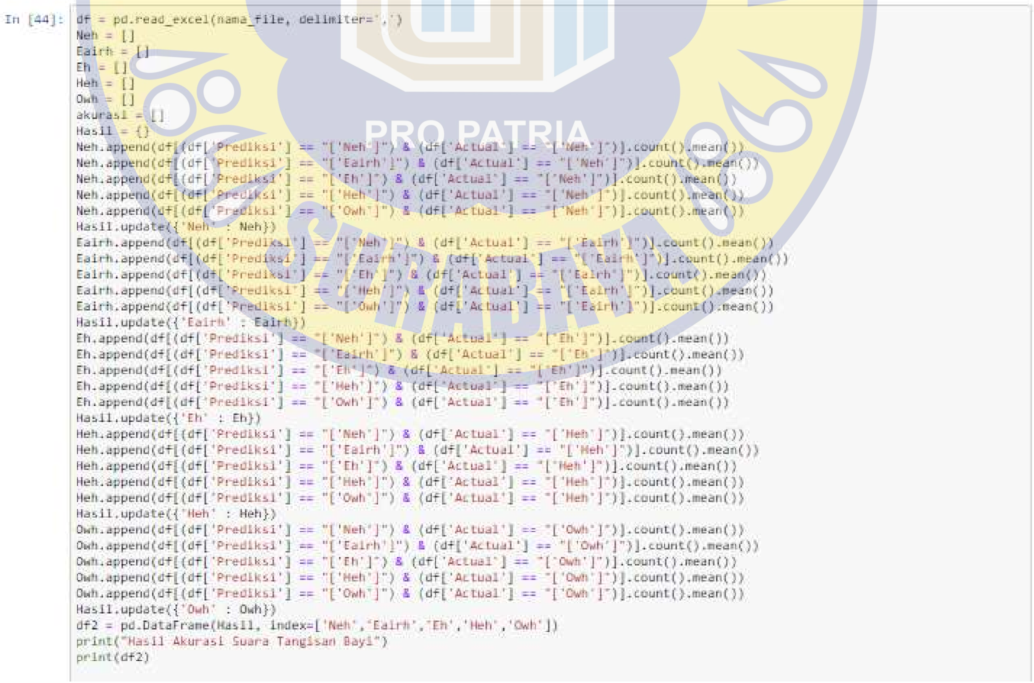
Proses machine learning



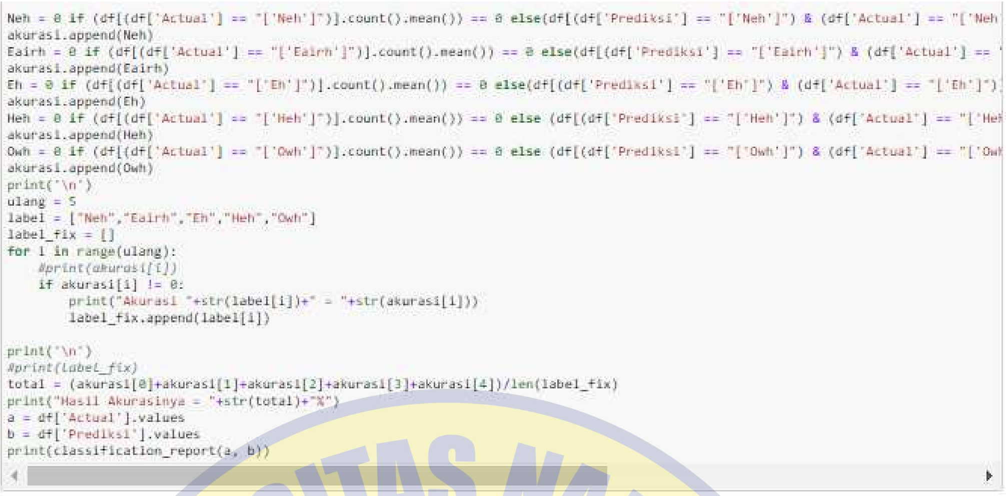
```
In [39]: k = 8
         loo = LeaveOneOut()
         loo.get_n_splits(x)
         scores_list = []
         y_test_list = []
         y_pred_list = []
         for i,j in loo.split(x):
             #print("TRAIN:", i, "TEST:", j)
             X_train, X_test = x[i], x[j]
             y_train, y_test = y[i], y[j]
             #print(X_train, X_test, y_train, y_test)
             knn = KNeighborsClassifier(n_neighbors=k)
             knn.fit(X_train,y_train)
             y_pred=knn.predict(X_test)
             scores = metrics.accuracy_score(y_test,y_pred)
             scores*=100
             scores_list.append(scores)
             y_test_list.append(y_test)
             y_pred_list.append(y_pred)

         df = pd.DataFrame({'akurasi': scores_list, 'Actual': y_test_list, 'Prediksi': y_pred_list})
         df.to_excel(nama_file,index=False)
```

proses menampilkan confussion matrix, akurasi, precision, recall



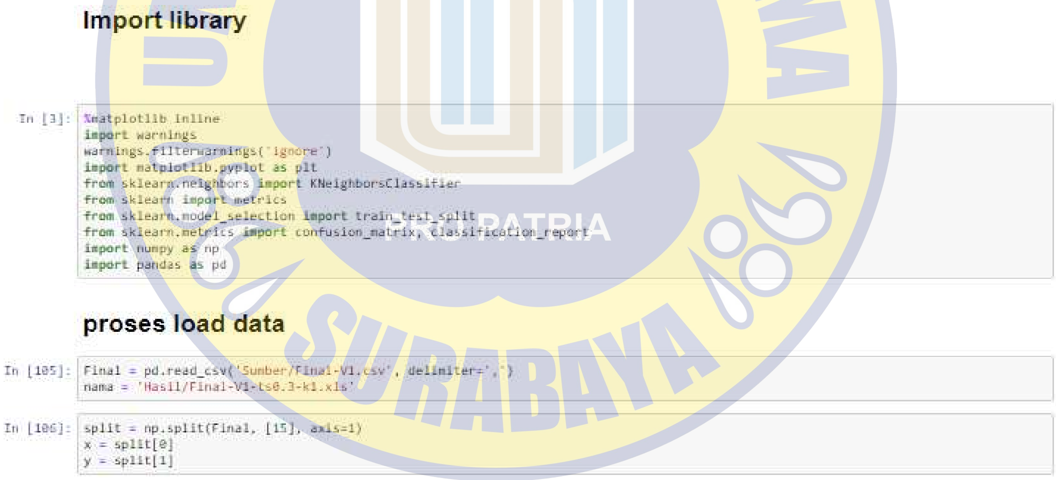
```
In [44]: df = pd.read_excel(nama_file, delimiter=',')
         Neh = []
         Eairh = []
         Eh = []
         Heh = []
         Owh = []
         akurasi = []
         Hasil = {}
         Neh.append(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Neh']")].count().mean())
         Neh.append(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == "['Neh']")].count().mean())
         Neh.append(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Neh']")].count().mean())
         Neh.append(df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Neh']")].count().mean())
         Neh.append(df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Neh']")].count().mean())
         Hasil.update({'Neh' : Neh})
         Eairh.append(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Eairh']")].count().mean())
         Eairh.append(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == "['Eairh']")].count().mean())
         Eairh.append(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Eairh']")].count().mean())
         Eairh.append(df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Eairh']")].count().mean())
         Eairh.append(df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Eairh']")].count().mean())
         Hasil.update({'Eairh' : Eairh})
         Eh.append(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Eh']")].count().mean())
         Eh.append(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == "['Eh']")].count().mean())
         Eh.append(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Eh']")].count().mean())
         Eh.append(df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Eh']")].count().mean())
         Eh.append(df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Eh']")].count().mean())
         Hasil.update({'Eh' : Eh})
         Heh.append(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Heh']")].count().mean())
         Heh.append(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == "['Heh']")].count().mean())
         Heh.append(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Heh']")].count().mean())
         Heh.append(df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Heh']")].count().mean())
         Heh.append(df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Heh']")].count().mean())
         Hasil.update({'Heh' : Heh})
         Owh.append(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Owh']")].count().mean())
         Owh.append(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == "['Owh']")].count().mean())
         Owh.append(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Owh']")].count().mean())
         Owh.append(df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Owh']")].count().mean())
         Owh.append(df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Owh']")].count().mean())
         Hasil.update({'Owh' : Owh})
         df2 = pd.DataFrame(Hasil, index=['Neh','Eairh','Eh','Heh','Owh'])
         print("Hasil Akurasi Suara Tangisan Bayi")
         print(df2)
```



```
Neh = 0 if (df[(df['Actual'] == "['Neh']")].count().mean()) == 0 else(df[(df['Prediksi'] == "['Neh']") & (df['Actual'] == "['Neh
akurasi.append(Neh)
Eairh = 0 if (df[(df['Actual'] == "['Eairh']")].count().mean()) == 0 else(df[(df['Prediksi'] == "['Eairh']") & (df['Actual'] == '
akurasi.append(Eairh)
Eh = 0 if (df[(df['Actual'] == "['Eh']")].count().mean()) == 0 else(df[(df['Prediksi'] == "['Eh']") & (df['Actual'] == "['Eh']")
akurasi.append(Eh)
Heh = 0 if (df[(df['Actual'] == "['Heh']")].count().mean()) == 0 else (df[(df['Prediksi'] == "['Heh']") & (df['Actual'] == "['Heh
akurasi.append(Heh)
Owh = 0 if (df[(df['Actual'] == "['Owh']")].count().mean()) == 0 else (df[(df['Prediksi'] == "['Owh']") & (df['Actual'] == "['Owh
akurasi.append(Owh)
print('\n')
ulang = 5
label = ["Neh","Eairh","Eh","Heh","Owh"]
label_fix = []
for i in range(ulang):
    #print(akurasi[i])
    if akurasi[i] != 0:
        print("Akurasi "+str(label[i])+" = "+str(akurasi[i]))
        label_fix.append(label[i])

print('\n')
#print(label_fix)
total = (akurasi[0]+akurasi[1]+akurasi[2]+akurasi[3]+akurasi[4])/len(label_fix)
print("Hasil Akurasinya = "+str(total)+"%")
a = df['Actual'].values
b = df['Prediksi'].values
print(classification_report(a, b))
```

Lampiran 3 : *Source Code K-Nearest Neighbours* dengan data sampling Percentage Rate



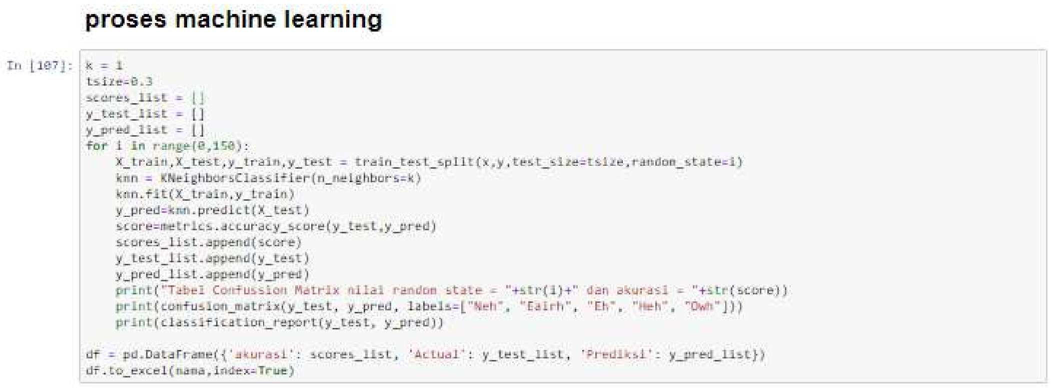
Import library

```
In [3]: %matplotlib inline
        import warnings
        warnings.filterwarnings('ignore')
        import matplotlib.pyplot as plt
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import metrics
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix, classification_report
        import numpy as np
        import pandas as pd
```

proses load data

```
In [105]: Final = pd.read_csv('Sumber/Final-V1.csv', delimiter=',')
          nama = 'Hasil/Final-V1-ts0.3-k1.xls'
```

```
In [106]: split = np.split(Final, [15], axis=1)
          x = split[0]
          y = split[1]
```

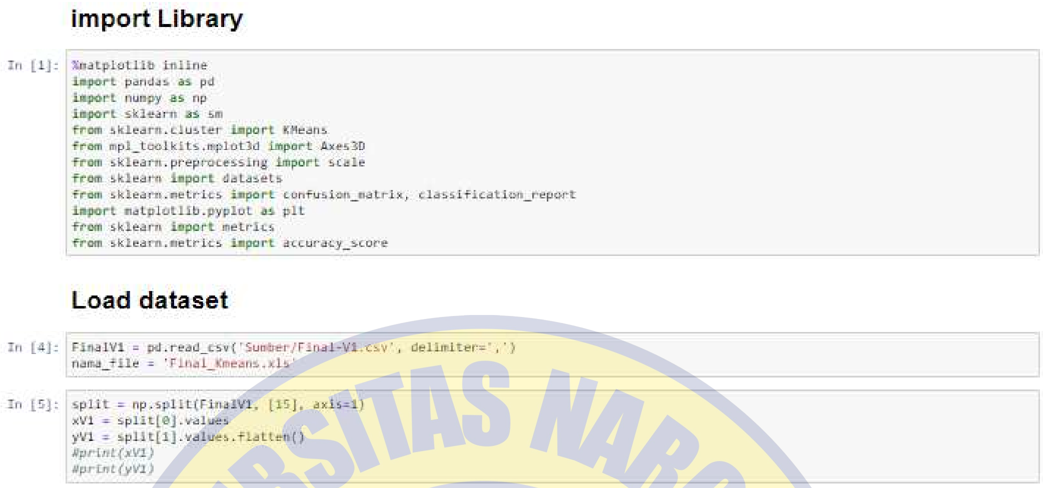


proses machine learning

```
In [107]: k = 1
          tsize=0.3
          scores_list = []
          y_test_list = []
          y_pred_list = []
          for i in range(0,150):
              X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=tsize,random_state=i)
              knn = KNeighborsClassifier(n_neighbors=k)
              knn.fit(X_train,y_train)
              y_pred=knn.predict(X_test)
              score=metrics.accuracy_score(y_test,y_pred)
              scores_list.append(score)
              y_test_list.append(y_test)
              y_pred_list.append(y_pred)
              print("Tabel Confussion Matrix nilai random state = "+str(i)+" dan akurasi = "+str(score))
              print(confusion_matrix(y_test, y_pred, labels=["Neh", "Eairh", "Eh", "Heh", "Owh"]))
              print(classification_report(y_test, y_pred))

          df = pd.DataFrame({'akurasi': scores_list, 'Actual': y_test_list, 'Prediksi': y_pred_list})
          df.to_excel(nama,index=True)
```

Lampiran 4: *Source Code K-Means*



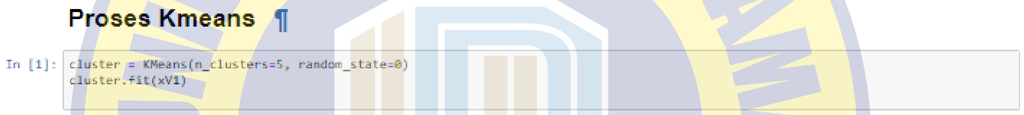
import Library

```
In [1]: %matplotlib inline
        import pandas as pd
        import numpy as np
        import sklearn as sm
        from sklearn.cluster import KMeans
        from mpl_toolkits.mplot3d import Axes3D
        from sklearn.preprocessing import scale
        from sklearn import datasets
        from sklearn.metrics import confusion_matrix, classification_report
        import matplotlib.pyplot as plt
        from sklearn import metrics
        from sklearn.metrics import accuracy_score
```

Load dataset

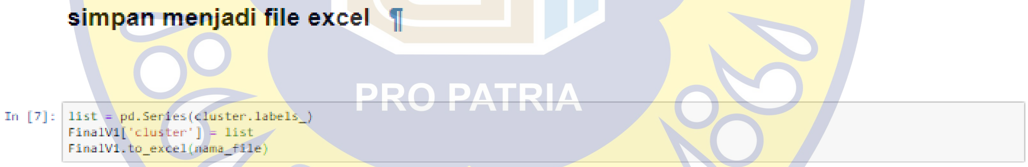
```
In [4]: FinalV1 = pd.read_csv('Sumber/Final-V1.csv', delimiter=',')
        nama_file = 'Final_Kmeans.xls'
```

```
In [5]: split = np.split(FinalV1, [15], axis=1)
        xV1 = split[0].values
        yV1 = split[1].values.flatten()
        #print(xV1)
        #print(yV1)
```



Proses Kmeans ¶

```
In [1]: cluster = KMeans(n_clusters=5, random_state=0)
        cluster.fit(xV1)
```



simpan menjadi file excel ¶

```
In [7]: list = pd.Series(cluster.labels_)
        FinalV1['cluster'] = list
        FinalV1.to_excel(nama_file)
```