Deniz Aytemiz

ECE5984-Homework 4

Part 1.

8. Choose the best value of k for each problem (binary and multiclass classification). Discuss your choice. Are they the same? Why or why not?

It can be observed for binary classification the best k value is 4. For multiple classification the best k value is 2. The algorithm works much better in terms of accuracy in binary classification. The reason for this might be that since there are less choices for binary classification it is more likely to make correct classifications.

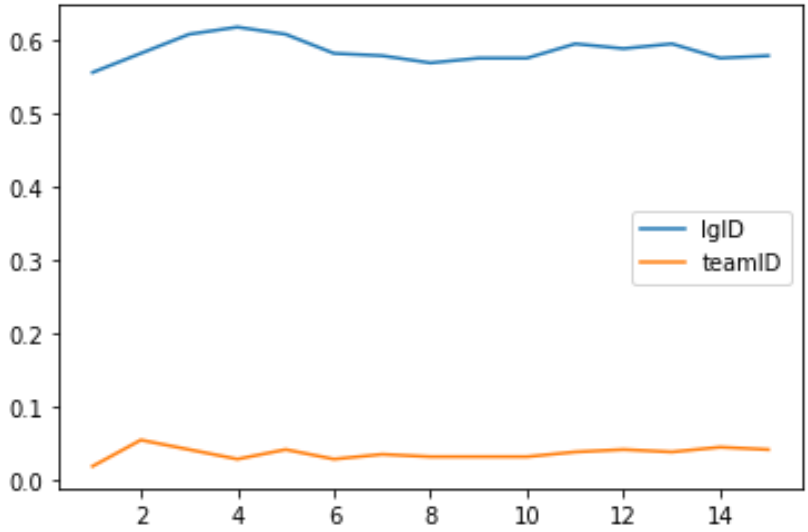

Figure 1: Accuracy graph for accuracy x k value when random seed is 22222

9. Repeat the process using a different random seed value. Are the best choices for k the same as before? Why or why not?

When random seed value changes the sequence of random value also changes. Therefore the training and test set also changes. This results in different accuracy values and different optimal k values. In this case the best k for binary classification is 11 and for multiple classifications it is 1.

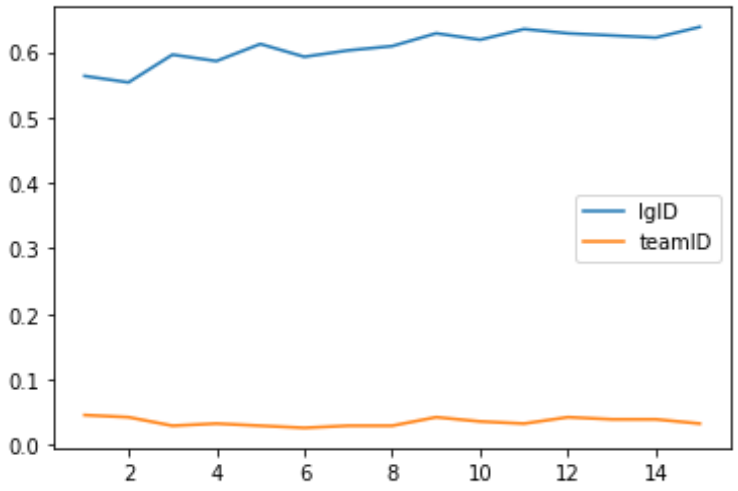

Figure 2: Accuracy graph with random seed is 2

Part 2:

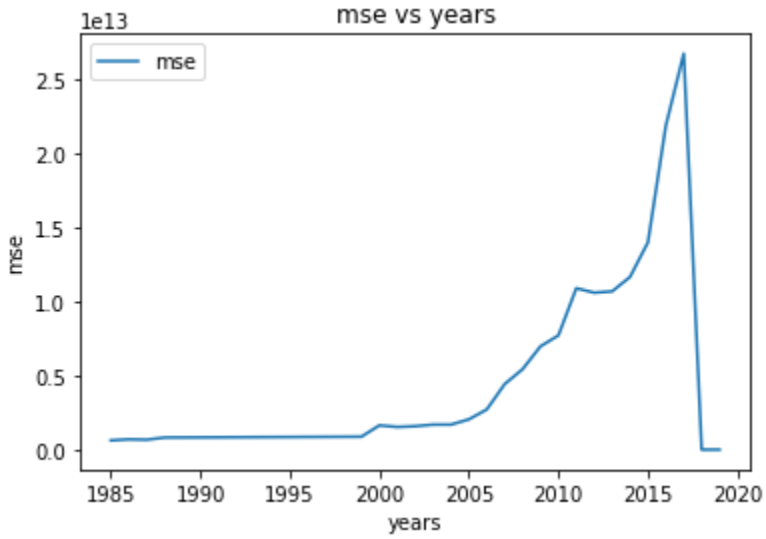

Figure 3: mean square error differing with respect to years

Code:

```
import numpy as np
import random
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

dataframe = pd.read_excel("BattingSalaries.xlsx");
df = dataframe[dataframe['yearID'] == 2016]

# Printing number of NaN values for numerical columns.
# It turns out except from salary all NaN values are in the same row.
# That is good since we can discard those rows all together.
print(df['CSrat'].isna().sum())
print(df['BBrat'].isna().sum())
print(df['SOrat'].isna().sum())
print(df['IBBrat'].isna().sum())
print(df['HBPrat'].isna().sum())
print(df['SHrat'].isna().sum())
print(df['SFrat'].isna().sum())
print(df['GIDPrat'].isna().sum())
print(df['yearPlayer'].isna().sum())
print(df['Salary'].isna().sum())

# deleting all the rows where CSrat is NaN valued
df = df[df['CSrat'].notna()]

# it turns out all NaN values are gone except from Salary column
print(df['CSrat'].isna().sum())
print(df['BBrat'].isna().sum())
print(df['SOrat'].isna().sum())
print(df['IBBrat'].isna().sum())
print(df['HBPrat'].isna().sum())
print(df['SHrat'].isna().sum())
print(df['SFrat'].isna().sum())
print(df['GIDPrat'].isna().sum())
print(df['yearPlayer'].isna().sum())
print(df['Salary'].isna().sum())
```

```
# since there are 355 rows with NaN valued row, instead of dropping them I
preferred to replace the
# NaN values with mean value instead
mean_value=df['Salary'].mean()
mean_value

df['Salary'].fillna(value=mean_value, inplace=True)

#checking whether there are still NaN values
print(df['Salary'].isna().sum())

# replaced string values with numerical ones for lgID column
df['lgID'] = df['lgID'].replace(['AL'],0)
df['lgID'] = df['lgID'].replace(['NL'],1)

print(df['teamID'].unique())

# replacing string values with numerical ones for teamID column
df['teamID'] = df['teamID'].replace(['MIN'],0)
df['teamID'] = df['teamID'].replace(['CHA'],1)
df['teamID'] = df['teamID'].replace(['NYA'],2)
df['teamID'] = df['teamID'].replace(['COL'],3)
df['teamID'] = df['teamID'].replace(['SLN'],4)
df['teamID'] = df['teamID'].replace(['CIN'],5)
df['teamID'] = df['teamID'].replace(['SFN'],6)
df['teamID'] = df['teamID'].replace(['CLE'],7)
df['teamID'] = df['teamID'].replace(['ARI'],8)
df['teamID'] = df['teamID'].replace(['TEX'],9)
df['teamID'] = df['teamID'].replace(['OAK'],10)
df['teamID'] = df['teamID'].replace(['PHI'],11)
df['teamID'] = df['teamID'].replace(['CHN'],12)
df['teamID'] = df['teamID'].replace(['HOU'],13)
df['teamID'] = df['teamID'].replace(['BAL'],14)
df['teamID'] = df['teamID'].replace(['LAA'],15)

df['teamID'] = df['teamID'].replace(['SDN'],16)
df['teamID'] = df['teamID'].replace(['LAN'],17)
df['teamID'] = df['teamID'].replace(['MIL'],18)

df['teamID'] = df['teamID'].replace(['MIA'],19)
```

```
df['teamID'] = df['teamID'].replace(['TBA'],20)
df['teamID'] = df['teamID'].replace(['SEA'],21)
df['teamID'] = df['teamID'].replace(['DET'],22)
df['teamID'] = df['teamID'].replace(['ATL'],23)

df['teamID'] = df['teamID'].replace(['TOR'],24)
df['teamID'] = df['teamID'].replace(['WAS'],25)
df['teamID'] = df['teamID'].replace(['PIT'],26)
df['teamID'] = df['teamID'].replace(['BOS'],27)
df['teamID'] = df['teamID'].replace(['NYN'],28)
df['teamID'] = df['teamID'].replace(['KCA'],29)

# for target value lgID, it is saved to variable y. All other numerical
values are placed within
# dataframe x
# ac_array's are constructed for holding accuracy values later on
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
sc = StandardScaler()
x = df.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer'],
axis=1)
y = df['lgID']
ac_array_lg = []
ac_array_team = []

# In the folllowing loop dataset is partioned for training and test parts.
Then they are normalized
# using standard scaler. For values of k from 1 to 15 model is created and
trained and accuracy of each model
# is saved to list ac_array_lg
from sklearn.neighbors import KNeighborsClassifier
for k in range(15):
 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
 X_train = sc.fit_transform(x_train)
 X_test = sc.transform(x_test)
 classifier = KNeighborsClassifier(n_neighbors = k+1, metric =
'minkowski', p = 2)
 classifier.fit(X_train, y_train)
```

```
 y_pred = classifier.predict(X_test)
 cm = confusion_matrix(y_test, y_pred)
 ac = accuracy_score(y_test,y_pred)
 print('k is', k+1)
 print('confusion matrix is', cm)
 print('accuracy is', ac)
 ac_array_lg.append(ac)

# doing the same process for target variable teamID
x = df.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer'],
axis=1)
y = df['teamID']

for k in range(15):
 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
 X_train = sc.fit_transform(x_train)
 X_test = sc.transform(x_test)
 classifier = KNeighborsClassifier(n_neighbors = k+1, metric =
'minkowski', p = 2)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 cm = confusion_matrix(y_test, y_pred)
 ac = accuracy_score(y_test,y_pred)
 print('k is', k+1)
 #print('confusion matrix is', cm)
 print('accuracy is', ac)
 ac_array_team.append(ac)

# creating the plot
k = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
plt.plot(k, ac_array_lg, label = "lgID")
plt.plot(k, ac_array_team, label = "teamID")
plt.title('accuracy vs k value')
plt.xlabel('k value')
plt.ylabel('accuracy')
plt.legend()
plt.show()

# doing the same process with different random seed number
```

```
x = df.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer'],
axis=1)
y = df['lgID']
ac_array_lg = []
ac_array_team = []

for k in range(15):
 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 2)
 X_train = sc.fit_transform(x_train)
 X_test = sc.transform(x_test)
 classifier = KNeighborsClassifier(n_neighbors = k+1, metric =
'minkowski', p = 2)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 cm = confusion_matrix(y_test, y_pred)
 ac = accuracy_score(y_test,y_pred)
 print('k is', k+1)
 #print('confusion matrix is', cm)
 print('accuracy is', ac)
 ac_array_lg.append(ac)


x = df.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer'],
axis=1)
y = df['teamID']
for k in range(15):
 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 2)
 X_train = sc.fit_transform(x_train)
 X_test = sc.transform(x_test)
 classifier = KNeighborsClassifier(n_neighbors = k+1, metric =
'minkowski', p = 2)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 cm = confusion_matrix(y_test, y_pred)
 ac = accuracy_score(y_test,y_pred)
 print('k is', k+1)
 #print('confusion matrix is', cm)
 print('accuracy is', ac)
```

```
 ac_array_team.append(ac)

# creating the pot for random seed = 2
k = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
plt.plot(k, ac_array_lg, label = "lgID")
plt.plot(k, ac_array_team, label = "teamID")
plt.title('accuracy vs k value')
plt.xlabel('k value')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

PART 2

```
# Load the dataset, and perform missing value processing
df = pd.read_excel("BattingSalaries.xlsx");
df = df[df['CSrat'].notna()]
mean_value=df['Salary'].mean()
df['Salary'].fillna(value=mean_value, inplace=True)

# one hot encoding for lgID AND teaMID
lg_dum = pd.get_dummies(df.lgID, prefix='lgID')
df = pd.concat([df, lg_dum], axis=1)
df

team_dum = pd.get_dummies(df.teamID, prefix='teamID')
df = pd.concat([df, team_dum], axis=1)
df

x =
df.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Salary'
], axis=1)
y = df['Salary']

from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
```

```
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

df = pd.read_excel("BattingSalaries.xlsx");
print(df['CSrat'].isna().sum())
print(df['BBrat'].isna().sum())
print(df['SOrat'].isna().sum())
print(df['IBBrat'].isna().sum())
print(df['HBPrat'].isna().sum())
print(df['SHrat'].isna().sum())
print(df['SFrat'].isna().sum())
print(df['GIDPrat'].isna().sum())
print(df['yearPlayer'].isna().sum())
print(df['Salary'].isna().sum())

df = df[df['CSrat'].notna()]
mean_value=df['Salary'].mean()
df['Salary'].fillna(value=mean_value, inplace=True)

print(df['CSrat'].isna().sum())
print(df['BBrat'].isna().sum())
print(df['SOrat'].isna().sum())
print(df['IBBrat'].isna().sum())
print(df['HBPrat'].isna().sum())
print(df['SHrat'].isna().sum())
print(df['SFrat'].isna().sum())
print(df['GIDPrat'].isna().sum())
print(df['yearPlayer'].isna().sum())
print(df['Salary'].isna().sum())

# creating dataframe for each year
df_1985 = df[df['yearID'] == 1985]
print(df_1985.size)
df_1986 = df[df['yearID'] == 1986]
```

```
print(df_1986.size)
df_1987 = df[df['yearID'] == 1987]
print(df_1987.size)
df_1988 = df[df['yearID'] == 1988]
print(df_1988.size)
df_1989 = df[df['yearID'] == 1989]
print(df_1989.size)
df_1990 = df[df['yearID'] == 1990]
print(df_1990.size)
df_1991 = df[df['yearID'] == 1991]
print(df_1991.size)
df_1992 = df[df['yearID'] == 1992]
print(df_1992.size)
df_1993 = df[df['yearID'] == 1993]
print(df_1993.size)
df_1994 = df[df['yearID'] == 1994]
print(df_1994.size)
df_1995 = df[df['yearID'] == 1995]
print(df_1995.size)
df_1996 = df[df['yearID'] == 1996]
print(df_1996.size)
df_1997 = df[df['yearID'] == 1997]
print(df_1997.size)
df_1998 = df[df['yearID'] == 1998]
print(df_1998.size)
df_1999 = df[df['yearID'] == 1999]
print(df_1999.size)
df_2000 = df[df['yearID'] == 2000]
print(df_2000.size)
df_2001 = df[df['yearID'] == 2001]
print(df_2001.size)
df_2002 = df[df['yearID'] == 2002]
print(df_2002.size)
df_2003 = df[df['yearID'] == 2003]
print(df_2003.size)
df_2004 = df[df['yearID'] == 2004]
print(df_2004.size)
df_2005 = df[df['yearID'] == 2005]
print(df_2005.size)
df_2006 = df[df['yearID'] == 2006]
```

```
print(df_2006.size)
df_2007 = df[df['yearID'] == 2007]
print(df_2007.size)
df_2008 = df[df['yearID'] == 2008]
print(df_2008.size)
df_2009 = df[df['yearID'] == 2009]
print(df_2009.size)
df_2010 = df[df['yearID'] == 2010]
print(df_2010.size)
df_2011 = df[df['yearID'] == 2011]
print(df_2011.size)
df_2012 = df[df['yearID'] == 2012]
print(df_2012.size)
df_2013 = df[df['yearID'] == 2013]
print(df_2013.size)
df_2014 = df[df['yearID'] == 2014]
print(df_2014.size)
df_2015 = df[df['yearID'] == 2015]
print(df_2015.size)
df_2016 = df[df['yearID'] == 2016]
print(df_2016.size)
df_2017 = df[df['yearID'] == 2017]
print(df_2017.size)
df_2018 = df[df['yearID'] == 2018]
print(df_2018.size)
df_2019 = df[df['yearID'] == 2019]
print(df_2019.size)

#for each year creating linear regression model
x =
df_1985.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1985['Salary']

print(y.isna().sum())

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
```

```
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1986.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1986['Salary']

print(y.isna().sum())

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1986.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1986['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))
```

```
print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1987.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1987['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1988.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1988['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
```

```
x =
df_1989.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1989['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1990.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1990['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1991.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1991['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1992.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1992['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1993.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1993['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1994.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1994['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1995.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1995['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
```

```
x =
df_1996.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1996['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1997.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1997['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1998.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1998['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_1999.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_1999['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2000.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2000['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2001.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2001['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2002.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2002['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
```

```
x =
df_2003.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2003['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2004.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2004['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2005.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2005['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2006.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2006['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2007.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2007['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2008.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2008['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2009.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2009['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
```

```
x =
df_2010.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2010['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2011.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2011['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2012.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2012['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2013.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2013['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2014.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2014['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2015.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2015['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2016.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2016['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
```

```
x =
df_2017.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2017['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2018.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2018['Salary']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))

#for each year creating linear regression model
x =
df_2019.drop(['playerID','yearID','stint','teamID','lgID','yearPlayer','Sa
lary'], axis=1)
y = df_2019['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state = 22222)
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
print(r2_score(y_test,y_pred))

print(mean_squared_error(y_test, y_pred))



mse=[623654994029.3147,694215210898.1947,
668939698084.1505,816607064039.7571,874374111994.0724,1637613358569.544,
1531749075973.9897, 1580288520180.2585, 1688685495759.4077
,1688685495759.4077, 2043991836545.8794, 2711100880035.7217,
4423922952125.537,5427953490555.42,  6971151099922.849, 7706113802219.227,
10888167869858.92, 10587392226384.525
, 10676125846121.3, 11649758306275.92,  13983045133973.732,
21850956782499.33, 26714913767919.824,0.0,0.0  ]
years=[1985,1986,1987,1988,1999,2000,2001,2002,2003,2004,2005,2006,2007,20
08,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019]

plt.plot(years, mse, label = "mse")
plt.title('mse vs years')
plt.xlabel('years')
plt.ylabel('mse')
plt.legend()
plt.show()
```