

Intern Report

Machine Learning and Tksolver

J Vishwanath | 8/6/2022

Contents

[WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES 2](#_Toc107425670)

[Week 1 3](#_Toc107425671)

[Introduction 3](#_Toc107425672)

[Understanding tksolver 3](#_Toc107425673)

[Understanding PCA 5](#_Toc107425674)

[Implementing PCA on python 7](#_Toc107425675)

[Linear regression 10](#_Toc107425676)

[Regression Implementation on Python 12](#_Toc107425677)

[TK solver Implementation using Python 14](#_Toc107425678)

[Logistic Regression 17](#_Toc107425679)

[TK Sovler Outputs 20](#_Toc107425680)

[TK Solver Implementation 23](#_Toc107425681)

[K-means 25](#_Toc107425682)

# WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES

|  |  |  |  |
| --- | --- | --- | --- |
| 1st Week | Date | Day | Topic |
| 7/6/2022 | Tuesday | Introduction |
| 8/6/2022 | Wednesday | Understanding TK Solver |
| 9/6/2022 | Thursday | Understanding Principal Component Analysis |
| 10/6/2022 | Friday | Implementing PCA |

|  |  |  |  |
| --- | --- | --- | --- |
| 2nd Week | Date | Day | Topic |
| 14/6/2022 | Tuesday | Machine Learning Basics |
| 15/6/2022 | Wednesday | Linear Regression and Implementation |
| 16/6/2022 | Thursday | Logistic Regression and Implementation |
| 17/6/2022 | Friday | KMeans and Implementation |

|  |  |  |  |
| --- | --- | --- | --- |
| 1st Week | Date | Day | Topic |
| 7/6/2022 | Tuesday | Introduction |
| 8/6/2022 | Wednesday | Understanding TK Solver |
| 9/6/2022 | Thursday | Understanding Principal Component Analysis |
| 10/6/2022 | Friday | Implementing PCA |

# Week 1

## Introduction

Bssic Orientation and Introduction of the company and what all to do.

## Understanding tksolver

TK Solver is a software for mathematical modeling and equation solving using a very unique declarative programming method while also providing procedural programming capabilities available in other languages like Python, Visual Basic, FORTRAN, C, C++, Pascal, etc.

Since 1982 many thousands of very satisfied customers have been using TK Solver in a wide range of industries such as:

Aerospace

Chemical and petrochemical

Automotive, agricultural and construction equipment manufacturing

Defense

R&D

Engineering colleges and universities – faculty, researchers and students.

Graphical user interface, application, table, Excel

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

## Understanding PCA

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Calendar

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, application, table, Excel

Description automatically generated

## Implementing PCA on python

A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, application, Teams

Description automatically generatedChart, scatter chart

Description automatically generatedGraphical user interface, text

Description automatically generated

## Linear regression

Main ideas behind the linear regression model:

1. Use least square to fit a line between the scatter points
2. Calculation of the R2
3. Calculation of p-value for the R2
4. When we fit a line through the scatter plot, the regression line represents the estimated job satisfaction for a given age.
5. However the real observation might not fall exactly on the regression line.
6. We try to explain the scatter plot with a linear equation of y = b0 + b1x.
7. The distance between the regression line and the data point represents the unexplained variation, which is also called the residual ei.
8. The method of least squares is used to minimize the residual.

Components of Linear Regression:

1. Regression Coefficient (or β1):

The Regression Coefficient in the above equation talks about the change in the value of dependent variable corresponding to the unit change in the independent variable. So, for e.g. if X1 increases or decreases by one unit, then Y will increase or decrease by β1 units. An important assumption followed by an ideal linear regression is that any increase or decrease in one independent variable will not have any corresponding changes in other independent variables.

2. Intercept (or β0):

Intercept is a constant value which tells us at what point in the x-y coordinate graph, should the regression line must start if it follows a linear regression. Since it is a constant value, hence it is not dependent on any change in independent variables. Even if the values of X=0, intercept will have a constant value. If the value of intercept is 0, that means, the line will start at the origin point (0,0).

3. Error Terms or Residuals (ϵ):

It is the difference between the actual and the predicted data point in the x-y coordinate graph

Chart, scatter chart

Description automatically generatedChart, line chart

Description automatically generated

### Regression Implementation on Python

Importing the required modules.

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

from sklearn import linear\_model

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generated

### TK solver Implementation using Python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import linear\_model

#sample dataset

df = pd.DataFrame({'area': [2600, 3000, 3200, 3600, 4000], 'prices': [

    550000, 565000, 610000, 680000, 725000]})

#specifing the train and test values

x\_train = df[['area']]

y\_train = df.prices

#Applying the linear regression model to find the best fit lines

reg = linear\_model.LinearRegression()

reg.fit(x\_train, y\_train)

#Line intercept

intre = reg.intercept\_

#R-square value of the line

r2\_score = reg.score(df[['area']], df.prices)

class FunctionInfo:

    def \_\_init\_\_(self):

        # Function information dictionary - { function name: [ list of arguments, return type, funtion description ] }

        self.func\_dict = {

            'TKPY\_linear\_regression\_main': [['float'], ['array'], ['LinearRegression Model']],

            'TKPY\_linear\_coef':[['int'],['float'],['linear coefficient of the line']],

            'TKPY\_linear\_intercept' : [['int'],['float'],['intercept on the y-axis']],

            'TKPY\_linear\_rscore' : [['int'],['float'],['accuracy of the model']]

        }

    def get\_func\_dict(self):

        return self.func\_dict

def TKPY\_linear\_intercept(a):

    return float(intre)

def TKPY\_linear\_rscore(a):

    return float(r2\_score)

def TKPY\_linear\_coef(a):

    coeff = reg.coef\_

    return(float(coeff))

def TKPY\_linear\_y\_pred(reg, a):

    y\_pred = reg.predict([[a]])

    return(float(y\_pred))

def TKPY\_linear\_regression\_main(a):

    pred = TKPY\_linear\_y\_pred(reg, a)

    coeff = TKPY\_linear\_coef(reg)

    return np.array([pred, r2\_score, coeff, intre])

Table

Description automatically generated with medium confidence

Graphical user interface, table

Description automatically generated

## Logistic Regression

It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable.

It gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. Graphical user interface

Description automatically generated with medium confidence

Table

Description automatically generated

Reading the dataframe and the items present inside

A screenshot of a computer screen

Description automatically generated with medium confidence

Splitting the train and test sample datasetA screenshot of a computer

Description automatically generated with medium confidence

Finding the accuracy, prediction probabilities and the predicted values of the testing dataset.A screenshot of a computer

Description automatically generated with medium confidence

### TK Sovler Outputs

Table

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Table

Description automatically generated

### TK Solver Implementation

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

import pandas as pd

import numpy as np

#prediciting whether whether a person will have insurance or not based on his age

a = pd.read\_csv(r"https://raw.githubusercontent.com/codebasics/py/master/ML/7\_logistic\_reg/insurance\_data.csv")

dfs = pd.DataFrame(a)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(dfs[['age']],dfs.bought\_insurance,train\_size=0.8,random\_state=100)

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X\_train, y\_train)

cmatrix = confusion\_matrix(y\_test, model.predict(X\_test))

class FunctionInfo:

    def \_\_init\_\_(self):

        # Function information dictionary - { function name: [ list of arguments, return type, funtion description ] }

        self.func\_dict =   {

                                'TKPY\_logistic\_main': [['float'], ['array'], ['reg Model']],

                                'TKPY\_logistic\_intercept' : [['float'],['float'],['intercept on Y-axis']],

                                'TKPY\_logistic\_coef' : [['float'],['float'],['Coefficient for the equation']],

                                'TKPY\_logistic\_acc' :[['float'],['float'],['accuracy of the given model']],

                                'TKPY\_logistic\_predict': [['float'],['float'],['predicted values for any value']],

                                'TKPY\_logistic\_confusionMatrix' : [['int'],['matrix'],["confusion\_matrix"]]

                            }

    def get\_func\_dict(self):

        return self.func\_dict

def TKPY\_logistic\_predict(age):

    pred =model.predict([[age]])

    return float(pred)

def TKPY\_logistic\_intercept(dummy):

    i= model.intercept\_

    return(float(i))

def TKPY\_logistic\_acc(dummy):

    Acc = model.score(X\_test,y\_test)

    return float(Acc)

def TKPY\_logistic\_coef(dummy):

    o= model.coef\_

    return(float(o))

def TKPY\_logistic\_confusionMatrix(a):

    cmatrix = confusion\_matrix(y\_test,model.predict(X\_test))

    cmatrix = cmatrix + a - a

    return np.array(cmatrix)

def TKPY\_logistic\_main(a):

    k=TKPY\_logistic\_intercept(model)

    l=TKPY\_logistic\_acc(model)

    s=TKPY\_logistic\_coef(model)

    return (np.array([ l,k,s]))

print(TKPY\_logistic\_coef(10))

## K-means

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

The following example shows unsupervised learning algorithm to predict the different income clusters of various individuals based on their age.

Chart, scatter chart

Description automatically generated

Finding the number of clusters using elbow plot

Graphical user interface

Description automatically generated

Scaling and transforming the data to as to make it more accurate. This is done because the there is a varying difference in the numeric values of age and salary.Chart

Description automatically generated

Import Kmeans from sklearn.clusterGraphical user interface, text

Description automatically generated

Plotting the cluster centers or centroids and finding the coordinates of the given centroids for various clusters

Graphical user interface

Description automatically generated

Predicting the right cluster for a datapoint

Text

Description automatically generated

### TKSolver Implementation

#Imported Libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

# Importing the the dataset from github

df = pd.read\_csv(r"https://raw.githubusercontent.com/codebasics/py/master/ML/13\_kmeans/income.csv")

df = pd.DataFrame(df)

#scaling the age and income values as both are numerically very different

scaler = MinMaxScaler()

stdf = scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])

stdf = scaler.fit(df[['Age']])

df['Age'] = scaler.transform(df[['Age']])

#creating the right number of clusters using Kmeans

km = KMeans(n\_clusters=3)

clust = km.fit\_predict(df[['Age','Income($)']])

cluster\_centers =km.cluster\_centers\_

class FunctionInfo:

    def \_\_init\_\_(self):

        # Function information dictionary - { function name: [ list of arguments, return type, funtion description ] }

        self.func\_dict =   {

                                'TKPY\_Clust\_Centers': [['float'], ['matrix'], ['cluster centers']],

                                'TKPY\_predVal' : [['float','float'],['float'],['Predict the cluster for a datapoint']],

                                'TKPY\_DataCluster': [['float'],['array'],['clusters in the dataset']],

                                "TKPY\_elbow\_plot" : [['float'],['array'],['elbow plot for clustering']],

                                "TKPY\_Kmeans\_Inertia":[['float'],['float'],['Intertia is the measure of well a dataset is clustered.']]

                            }

    def get\_func\_dict(self):

        return self.func\_dict

# a is a dummy value, ignore it!!

# + a - a has been done so as to avoid the error of generating random numbers in the tksolver as it requires for the use of dummy variables in one way or another.

#differentaion and categorizing various datapoints into different clusters i.e 0,1,2

def TKPY\_DataCluster(a):

    clusters = clust + a -a

    return  np.array(clusters)

# Predicting the cluster for a specific coordinate

def TKPY\_predVal(a,b):

    pred= km.predict([[a,b]])

    return(float(pred))

#coordinates of various centeroids formed

def TKPY\_Clust\_Centers(a):

    points = cluster\_centers + a-a

    return np.array(points)

#elbow plot to determine the number of clusters to choose from and run the algorithim

def TKPY\_elbow\_plot(a):

    sse = []

    k\_rng = range(1,10)

    for k in k\_rng:

        km = KMeans(n\_clusters=k)

        km.fit(df[['Age','Income($)']])

        sse.append(km.inertia\_)

    sse\_array = np.array(sse) + a - a

    return sse\_array

#calculating the best fit inertia i.e Least sum of squared errors

def TKPY\_Kmeans\_Inertia(a):

    return np.float(km.inertia\_)