ITCS159 Software Lab for Basic Scientific Problem Solving
Lab 12: Introduction to data science using case studies in Kaggle (cont.)

November 1, 2021

Objective

This lab is the next steps of data analytics on the HR data sets continuing from the previous lab (i.e. Lab 11). This lab focus on developing a prediction model using machine learning techniques such as Random Forests. You are expected to sense the common processes in order to develop the predictive model. You do not have to worry about how algorithms work. Next, you will continue your analysis on the HR data.

Turnover V.S. AverageMonthlyHours

The following code draws the figure compared between the turnover rate and working hours on average across a month. You should try to interpret the figure to answer the questions below.

```
fig = plt.figure(figsize=(15,4))
ax=sns.kdeplot(df.loc[(df['turnover'] == 0),'
    averageMonthlyHours'] , color='b',shade=True, label='no
    turnover')
ax=sns.kdeplot(df.loc[(df['turnover'] == 1),'
    averageMonthlyHours'] , color='r',shade=True, label='
    turnover')
ax.set(xlabel='Employee Average Monthly Hours', ylabel='
    Frequency')

plt.title('Employee AverageMonthly Hours Distribution -
    Turnover V.S. No Turnover')
```

From the graph, there are two groups of employees who left the company: had less hours of work and had too many hours of work. Can you give a suggestion on which group tend to leave the company? How should we deal with them.

Turnover V.S. Satisfaction

The next aspect that we need to analyze is about the satisfaction level of employees. The employee's happiness might correlate with the turn over rate that we are exploring. Let use the following code to plot a graph compared between the turnover and the satisfaction level to answer the questions.

```
fig = plt.figure(figsize=(15,4))
ax=sns.kdeplot(df.loc[(df['turnover'] == 0),'satisfaction'] ,
    color='b',shade=True, label='no turnover')
ax=sns.kdeplot(df.loc[(df['turnover'] == 1),'satisfaction'] ,
    color='r',shade=True, label='turnover')

plt.title('Employee Satisfaction Distribution - Turnover V.S.
    No Turnover')
```

Which groups of employees tend to leave the company (low satisfaction level, medium satisfaction level, or high satisfaction level)? Can you guess that why employees with the high satisfaction level still left the company?

Satisfaction V.S. Evaluation

To answer the previous question that why employees with the high satisfaction level left the company, we might need to look at different aspects. Now, you explore the evaluation results to further analyze why they left the company. Let use the following code to plot a figure compared between the satisfaction level and the evaluation result.

```
sns.lmplot(x='satisfaction', y='evaluation', data=df, fit_reg=
    False,  hue='turnover')
```

How many clusters for employees who left the company? Which clusters show the hard-working and sad employee?

Milestone 1: You have now completed the Milestone 1. Keep going!

Using K-Means Clustering to analyze

As you studied K-Means Clustering from the previous lab, let use it for our analysis. The following code applies K-Means on the satisfaction level and the evaluation result to cluster the data into 3 clusters.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3,random_state=2)
kmeans.fit(df[df.turnover==1][["satisfaction","evaluation"]])

kmeans_colors = ['green' if c == 0 else 'blue' if c == 2 else '
    red' for c in kmeans.labels_]

fig = plt.figure(figsize=(10, 6))
plt.scatter(x="satisfaction",y="evaluation", data=df[df.
    turnover==1], alpha=0.25,color = kmeans_colors)
plt.xlabel("Satisfaction")
plt.ylabel("Evaluation")
plt.scatter(x=kmeans.cluster_centers_[:,0],y=kmeans.
    cluster_centers_[:,1],color="black",marker="X",s=100)
plt.title("Clusters of Employee Turnover")
plt.show()
```

However, this is only one aspect. Can you try to apply K-Means on others e.g. project count V.S. average monthly hours, work accident V.S. satisfaction, and project count V.S. promotion to gain more insight?

Milestone 2: You have now completed the Milestone 2. Keep going!

Apply machine learning techniques to build a prediction model

In this exercise, you explore the process of adopting a machine learning technique (i.e. decision tree) to build a prediction model. Decision Tree is a supervised learning algorithm that can be represented as a tree graph model consisted of decision nodes and leaf nodes represented the target output (e.g. category). A decision node is a rule that is derived based on a feature that can split most data into their actual category. The classification of an unknown data is achieved by passing it through the tree starting at the top and moving down until a leaf node is reached. The value at that leaf node gives the predicted output for the data. At each node, the branch is selected based on the value of the corresponding feature. The significant benefit of decision tree classifier is that it offers an explainable model. Thus, most of the work that focuses on interpretable models selected this technique.

The following code is used to build the decision tree based on our HR data to predict the turn over.

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (12,6)

# you can skip this part, if you already changed the column
    name.
df = df.rename(columns={'satisfaction_level': 'satisfaction', '
    last_evaluation': 'evaluation', 'number_project': '
    projectCount', 'average_montly_hours': 'averageMonthlyHours'
    , 'time_spend_company': 'yearsAtCompany', 'Work_accident': '
    workAccident', 'promotion_last_5years': 'promotion', 'sales'
     : 'department', 'left' : 'turnover'
})

# Convert these variables into categorical variables
df["department"] = df["department"].astype('category').cat.
    codes
df["salary"] = df["salary"].astype('category').cat.codes

# Create train and test splits
target_name = 'turnover'
```

```
X = df.drop('turnover', axis=1)

y=df[target_name]

X_train, X_test, y_train, y_test = train_test_split(X,y,
    test_size=0.15, random_state=123, stratify=y)

dtree = tree.DecisionTreeClassifier(max_depth=3, class_weight="
    balanced", min_weight_fraction_leaf=0.01)
dtree = dtree.fit(X_train,y_train)
```

Now, **dtree** is your trained model. As mentioned, decision tree is interpretable model. Next, you are going to plot a tree structure derived from your model using the following code.

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz

import graphviz
import pydotplus

dot_data = StringIO()
export_graphviz(dtree, out_file=dot_data,
filled=True, rounded=True,
special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

dotfile = open("dtree2.dot", 'w')
tree.export_graphviz(dtree, out_file = dotfile, feature_names =
     X.columns)
dotfile.close()
```

Note that you may need to install new packages e.g. pydotplus, please refer to the previous lab of how to install new packages on Conda. However, if the command `conda install` does not work, please use `pip install`.

You will get the file called "dtree2.dot". Go to `http://webgraphviz.com/` and paste the content stored in the dtree2.dot file to the textbox to generate the tree graph. Can you try to interpret the result?

Now, let try to change the parameters of the decision tree at this line, and see whether the tree graph structure changes.

```
dtree = tree.DecisionTreeClassifier(max_depth=3, class_weight="
    balanced", min_weight_fraction_leaf=0.01)
```

Milestone 3: You have now completed the Milestone 3. Raise YOUR HAND NOW!