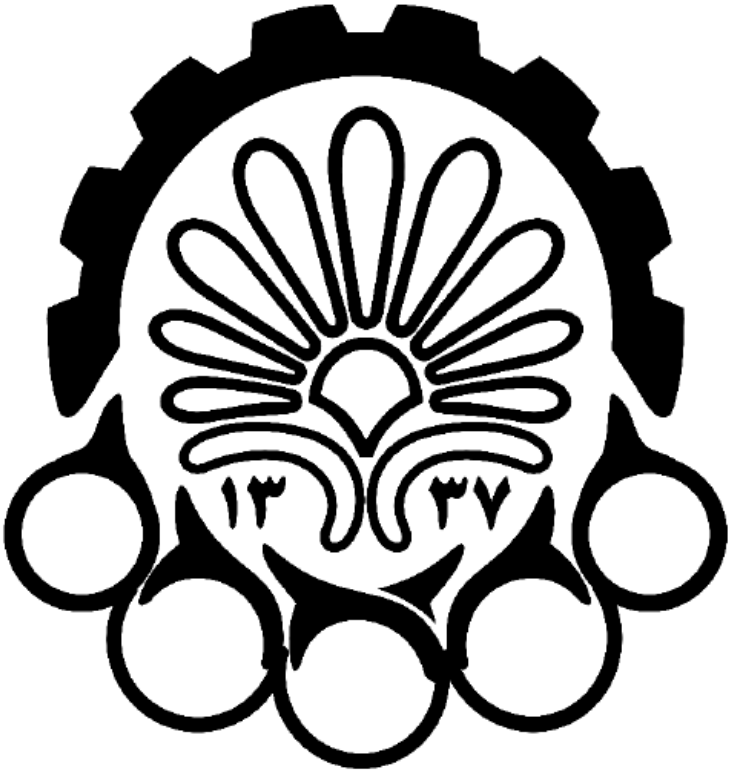

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

گزارش تمرین سوم داده کاوی

محبوبه شاکری ۹۵۳۱۳۰۱

استاد: دکتر امیر مزلقانی

زمستان ۹۹

سوال ۸ :

Random forest:

در ابتدا داده های غیر عددی مانند جنسیت به داده های عددی (صفر و یک) تبدیل میکنیم و همینطور مقادیر null برای ویژگی های fare و Age به ترتیب با میانه و میانگین داده ها پر میکنیم.

```
[ ]  data = pd.read_csv("train.csv")

     # preproccessing
     data["Fare"]=data["Fare"].fillna(data["Fare"].dropna().median())
     data["Age"]=data["Age"].fillna(data["Age"].dropna().mean())
     data.loc[data["Sex"]=="male","Sex"]=0
     data.loc[data["Sex"]=="female","Sex"]=1
     data["Embarked"]=data["Embarked"].fillna("5")
     data.loc[data["Embarked"]=="S","Embarked"]=0
     data.loc[data["Embarked"]=="C","Embarked"]=1
     data.loc[data["Embarked"]=="Q","Embarked"]=2
```

سپس ویژگی هایی که مناسب برای آموزش بودن را انتخاب کردیم و داده ها را به دو دسته ی آموزش و تست به نسبت ۸ به ۲ تقسیم میکنیم.

```
feature_names1 =["Pclass","Sex","Age","Fare","SibSp" ,"Parch","Embarked"]
x = data[feature_names1].values
y = data["Survived"].values

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=5)
```

سپس برای آموزش دادن جنگل ها تصادفی مختلفی را با حالات مختلف آموزش میدهیم (۲ حالت برای عمق درخت ها و ۲ حالت برای تابع تقسیم gini یا entropy در مجموع ۴ حالت مختلف). و نتایج در زیر آمده است.

```
random_forest1 = RandomForestClassifier(n_estimators=100,criterion="gini",max_depth=7)
random_forest1.fit(X_train, Y_train)

Y_prediction = random_forest1.predict(X_test)

random_forest1.score(X_train, Y_train)

print("Accuracy for 7 features with gini and max_depth=7 on train:", random_forest1.score(X_train, Y_train))
print("Accuracy for 7 features with gini and max_depth=7 on test:",  metrics.accuracy_score(Y_test, Y_prediction))
```

نتایج:

```
Accuracy for 7 features with gini and max_depth=7 on train: 0.8918539325842697
Accuracy for 7 features with gini and max_depth=7 on test: 0.8491620111731844
```

Gini and max depth=7

```
Accuracy for 7 features with gini and max_depth=2 on train: 0.7921348314606742
Accuracy for 7 features with gini and max_depth=2 on test: 0.8100558659217877
```

Gini and max depth=2

```
Accuracy for 7 features with entropy and max_depth=7 on train: 0.8904494382022472
Accuracy for 7 features with entropy and max_depth=7 on test: 0.8435754189944135
```

Entropy and max depth=7

```
Accuracy for 7 features with entropy and max_depth=2 on train: 0.8202247191011236
Accuracy for 7 features with entropy and max_depth=2 on test: 0.8379888268156425
```

Entropy and max depth=2

۱- و همینطور که مشخص است دقت در روش جنگل تصادفی از درخت تصمیم بیشتر است.

۲- با مقایسه ی زمان دو روش همانطور که انتظار میرفت روش جنگل تصادفی سرعت یادگیری و تست کمتری نسبت به درخت تصمیم دارد.

```python
import time
#Calculating Time
start = time.time()

random_forest1 = RandomForestClassifier(n_estimators=100,criterion="gini",max_depth=7)
random_forest1.fit(X_train, Y_train)

Y_prediction = random_forest1.predict(X_test)

random_forest1.score(X_train, Y_train)
end = time.time()

print(f"Runtime for random forest is {end - start}")

start = time.time()
dt11 = DecisionTreeClassifier(random_state=5,max_depth=7,criterion="entropy")
dt11=dt11.fit(X_train,Y_train)
y_pred = dt11.predict(X_test)

end = time.time()
print(f"Runtime for decision tree is {end - start}")
```

```
Runtime for random forest is 0.18016290664672852
Runtime for decision tree is 0.002962350845336914
```

سوال ۹ :

SVM:

۱- در ابتدا داده های غیر عددی مانند جنسیت به داده های عددی (صفر و یک) تبدیل میکنیم و همینطور مقادیر null برای ویژگی های fare و Age به ترتیب با میانه و میانگین داده ها پر میکنیم.

```
[ ]  data = pd.read_csv("train.csv")

     # preproccessing
     data["Fare"]=data["Fare"].fillna(data["Fare"].dropna().median())
     data["Age"]=data["Age"].fillna(data["Age"].dropna().mean())
     data.loc[data["Sex"]=="male","Sex"]=0
     data.loc[data["Sex"]=="female","Sex"]=1
     data["Embarked"]=data["Embarked"].fillna("5")
     data.loc[data["Embarked"]=="S","Embarked"]=0
     data.loc[data["Embarked"]=="C","Embarked"]=1
     data.loc[data["Embarked"]=="Q","Embarked"]=2
```

سپس ویژگی هایی که مناسب برای آموزش بودن را انتخاب کردیم و داده ها را به دو دسته ی آموزش و تست به نسبت ۸ به ۲ تقسیم میکنیم.

```
feature_names1 =["Pclass","Sex","Age","Fare","SibSp" ,"Parch","Embarked"]
x = data[feature_names1].values
y = data["Survived"].values

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=5)
```

۲- سپس svm را با هسته ی خطی آموزش داده ایم.

```
#SVM linear
from sklearn import svm

svclassifier = svm.SVC(kernel='linear')
svclassifier.fit(X_train, Y_train)

Y_prediction = svclassifier.predict(X_test)

print("Accuracy linear kernel on train:", svclassifier.score(X_train, Y_train))
print("Accuracy linear kernel on test:",  metrics.accuracy_score(Y_test, Y_prediction))
```

دقت به دست آمده به صورت زیر میباشد.

```
Accuracy linear kernel on train: 0.7851123595505618
Accuracy linear kernel on test: 0.7932960893854749
```

۳- همانطور که مشخص است دقت در جنگل تصادفی بیشتر است زیرا در جنگل تصادفی داده به صورت مخلوطی از داده های عددی و غیر عددی هستند بهتر پاسخ میدهد در حالی که در svm خطی با استفاده از فاصله این پاسخ سنجیده میشود و همینطور در داده های بیشتر از دو کلاسه جنگل تصادفی پاسخ بهتری میدهد و همینطور نسبت به outlier ها مقاومت تر از svm است.

۴- مدل را با rbf آموزش میدهیم.

```
#SVM rbf
svclassifier2 = svm.SVC(kernel='rbf' ,C=1E3)
svclassifier2.fit(X_train, Y_train)

Y_prediction = svclassifier2.predict(X_test)

print("Accuracy linear kernel on train:", svclassifier2.score(X_train, Y_train))
print("Accuracy linear kernel on test:",  metrics.accuracy_score(Y_test, Y_prediction))
```

نتیجه :

```
Accuracy linear kernel on train: 0.827247191011236
Accuracy linear kernel on test: 0.8379888268156425
```

۵- همینطور که مشخص است مقدار دقت افزایش یافته است. به این دلیل که هسته ی خطی برای داده هایی که به صورت خطی قابل جداسازی هستند بهتر پاسخ میدهد. و در این مساله با در نظر گرفتن ویژگی ها استفاده از هسته ی غیر خطی بهتر است.