Job Change Of Data Scientist Project

Ahmet Selim Pehlivan

August 10, 2022

Abstract

Projede data scientist rolündeki çalışanlar arasında iş değiştirmek isteyenleri tespit eden bir model geliştirdim. Geliştirdiğim model, insan kaynakları işe alım departmanlarınca kullanılabilir. Bu proje sayesinde HR verilerinde sınıflandırıcı modellerinin verdiği sonuçlar gözlemledim. Proje veri analizi, veri işlenmesi, model eğitimi, eğitilen modellerin karşılaştırılması ve sonuç kısımlarını içermektedir.

1 Giriş

Job Change Of Data Scientist Project temelinde iş değiştirmek isteyenleri ve istemeyenleri tespit edeceğimiz bir classification problemidir. Projedeki gemel amacımız boş verilerin bulunduğu ve feature'ların düzgün olmadığı bu veri setini iyi öğrenebilecek bir model geliştirmekti. Daha önceki çalışmalarda "%80" accuracy'e ulaşmıştır. Bu projenin amacı farklı classification modelleri deneyerek ve etkili veri mühendisliği adımlarıyla bu accuracy skorunu aşmaktı. "%80" accuracy skorunu aşarak "%85" gibi diğer çalışmaların önünde bir skor elde etmeyi amaçladım.

2 Literatür Araştırması

Benzer projelerde nominal veriler için one-hot encoding kullanılarak verileri arasında hiyerarşik bir durumun oluşmasının önüne geçilirken, ordinal veriler için bu hiyerarşinin korunması adına bu ordinal-encoder kullanılıyor. Okuduğum bazı medium makaleleri de bunu destekler nitelikteydi. Benzer projelere baktığımda model seçimi olarak decision tree, svm classifier, gradientboostclassifier, adaboost, xgboost, catboost kullanıldığını gördüm. Özellikle kayıp verileri bulma, onlara değer vermek ve başarımı artırma gibi özelliklerinden dolayı xgboost ve catboost çok fazla kullanılıyor. Bu yöntemlerle az veri varken de başarım artabiliyor fakat ensemble modellerin daha iyi sonuç vermesi için fazla veri gerekiyor. Yaptığım literatür taramlarıyla xgboost ve catboost gibi decision tree temelli modellerin Regularizasyon, missing value sorunu çözümü ve cross-validation'ı kendisi modelin içerisinde yaptığını ve ekstradan bunlarla uğraşmak zorunda kalınmadığını gördüm. Ayrıca benzer çalışmalarda boosting modellerinin hyperparametreleri'ni veri setine en uygun şekilde seçebilmek için optimizasyon algroitmaları kullanılıyor. Bu yaklaşımları projemde deneyip farkları tespit edip, başarımı artırmak için kullandım.

3 Veri Seti, Veri Özellikleri, Öznitelikleri

3.1 Veri Seti

Projedeki veri seti train ve test olmak üzere iki gruba ayrılmış durumda. Train verilerinde 19158 sample ve 13 feature bulunuyor. Bu featurelardan bazıları kendileri arasında bir ilişkiye sahip. Veri setinde fazla sayıda da boş veri de bulunuyor. Test verisinde ise 2130 sample bulunuyor. Train veri setinde olduğu gibi test verisinde de boş değerler bulunuyor. Veri setine baktığımızda target değerlerin karşılaştırılması sonucu veri setinin imbalanced bir yapıda olduğunu görüyoruz.

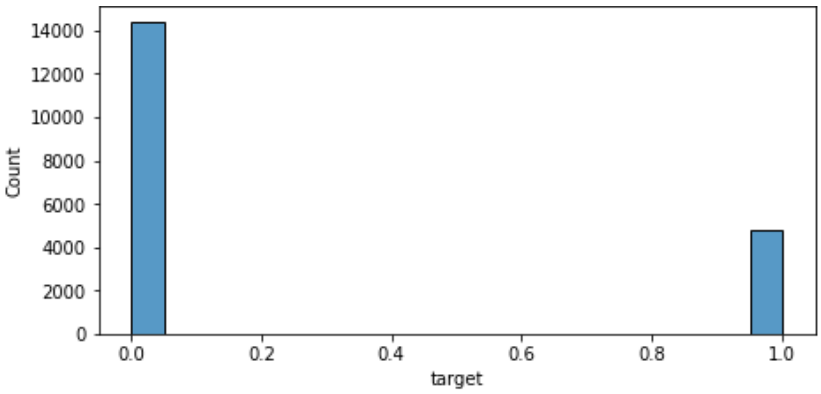


Figure 1: Target Value

3.2 Öznitelikler

Feature'lardan enrollee id feature'ı herhangi bir anlama sahip olmayan sadece iş başvurusu için kaydolanlara atanmış numaralardır. Bu feature'ı veri setinden sildim. Diğer feature'lar içerisindeki;

3.2.1 Kategorik Veriler

City, gender, relevent experience, enrolled university, education level, major discipline, company type last new job

Veri setindeki kategorik verileri sayısal verilere çevirmek için ordinal verilerde ordinal encoding ve nominal veriler için ise one-hot-encoding yaklaşımını kullandım.

Veri Setindeki Ordinal Veriler:

- relevent experience
- enrolled university
- education level
- company size
- company type

Veri Setindeki Nominal Veriler:

- City
- Gender
- Major discipline

Veri setindeki kategorik nominal verileri numerik hale getirmek için one hot encoding kullandım. Gender ve major discipline feature'larını one hot işleminden rahatlıkla geçirdim. Fakat city feature'ının çok fazla değeri olduğu için başka bir yöntem kullanmam gerekliliği ortaya çıktı.

3.2.2 Sayısal Veriler

- training hours
- city development index
- experience
- company size
- last new job

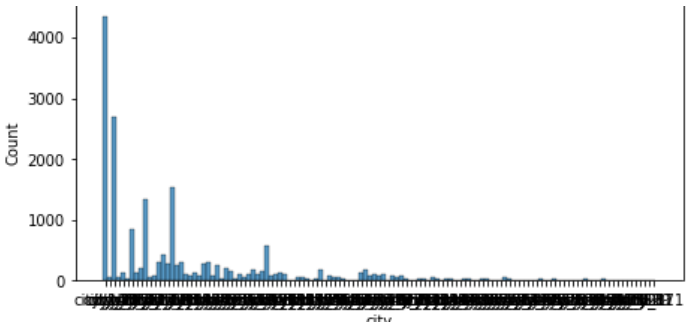


Figure 2: City Feature

3.3 Pre Processing

3.3.1 Nominal Verilerin Numerik Hale Getirilmesi

Figure 2'de City feature'unda one hot encoding öncesi farklı bir işlem ile farklı sample sayısının düzenlenmesi gerekiyor.

Öncelikle city ve city development index değerlerinden oluşan ayrı bir veri seti oluşturdum. Bu iki feature'un birbiri ile ilişkili olmasından dolayı feature'ları birlikte kullandım. Bu veri setine elimizdeki target ile K-Means clustering uyguladım. Bu K-means sayesinde 150 farklı city feature'ımı target'a göre 5 adet cluster'a map etmiş oldum. predict ettiğim değerleri aldım ve veri setimdeki city feature'ı ile değiştirdim. Böylelikle 5 ayrı değerden oluşan bir city feature'ı elde etmiş oldum. Daha sonra yeni elde ettiğim city feature'ıma de one hot encoding uyguladım. Böylelikle veri setimdeki feature sayısı 24'e çıktı. Böylelikle az da olsa boyut artmasına rağmen veriler arasındaki hiyerarşi olmadan bu featureları değerlendirmiş oldum.

3.3.2 Ordinal Verilerin Numerik Hale Getirilmesi

Ordinal veriler için ise ordinal encoding kullandım ve verileri anlamlı hiyerarşik bir biçimde numerik hale getirdim. Bu işlem elimdeki tüm veri seti numerik hale gelmiş oldu.

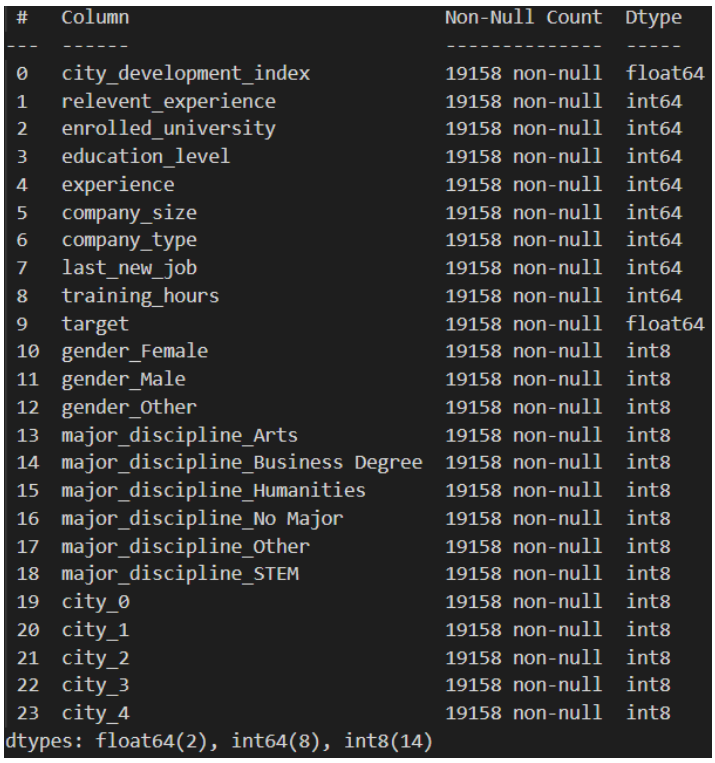
3.3.3 Boş Verilerin Doldurulması

Boş verilerin doldurulması için üç farklı yöntem kullandım.

- Ortalama değer ile doldurmak Bu yaklaşım sonrası gender, enrolled university gibi kategorik verilerde boş değerler sürekli aynı değerler ile doluyordu. Bu sebeple verinin dağılımı ortalama kategorik verilerin ağırlığına doğru kayıyor ve outlier'lar oluşuyordu. Bu sebeple bu yöntemden vazgeçtim.
- Rastgele değere göre doldurmak Rastgele değerlerle doldurmak ortalamaya göre daha iyi bir sonuç vermesine karşın knn'e karşı daha düşük accuracy sonucu elde etmeme sebep oldu.
- KNN ile en yakın değere göre doldurmak Bu yaklaşımda ise tüm veriyi KNN ile boş değerleri en yakın komşunun değerine göre doldurdum ve az da olsa accuracy'de iyi bir sonuç elde ettim.

3.3.4 Veri Setinin Balanced Hale Getirilmesi

Veri setini balanced hale getirmek için ise imbalanced data problemini çözen bilindik bir oversampling methodu olan SMOTE kullandım. Bu sayede veriyi daha balanced hale getirdim. Ayrıca training verim 19 bin'den 22'bine çıktı. Az veri ile training yapmamak için veri setini küçültmek yerine oversampling yaptım. Böylece daha balanced bir veri seti elde etmiş oldum.



#	Column	Non-Null Count	Dtype
0	city_development_index	19158 non-null	float64
1	relevent_experience	19158 non-null	int64
2	enrolled_university	19158 non-null	int64
3	education_level	19158 non-null	int64
4	experience	19158 non-null	int64
5	company_size	19158 non-null	int64
6	company_type	19158 non-null	int64
7	last_new_job	19158 non-null	int64
8	training_hours	19158 non-null	int64
9	target	19158 non-null	float64
10	gender_Female	19158 non-null	int8
11	gender_Male	19158 non-null	int8
12	gender_Other	19158 non-null	int8
13	major_discipline_Arts	19158 non-null	int8
14	major_discipline_Business Degree	19158 non-null	int8
15	major_discipline_Humanities	19158 non-null	int8
16	major_discipline_No Major	19158 non-null	int8
17	major_discipline_Other	19158 non-null	int8
18	major_discipline_STEM	19158 non-null	int8
19	city_0	19158 non-null	int8
20	city_1	19158 non-null	int8
21	city_2	19158 non-null	int8
22	city_3	19158 non-null	int8
23	city_4	19158 non-null	int8

dtypes: float64(2), int64(8), int8(14)

Figure 3: Features

3.3.5 Normalization

VSon olarak eğitime geçmeden önce veriyi normalize ettim. Normalizasyon için StandardScalar kullandım. Böylelikle elimdeki verileri mean'den çıkartıp ve standart sapmasına bölmüş oldum. Normalizasyon yapmaya ihtiyaç duydum çünkü verideki çoğu değer binary dağılırken, training hours ve experience verileri 0-20 ve 0-100 arasında bir dağılıma sahiplerdi.

4 Eğitim

Eğitim için classification methodlarını kullandım. Bu veri seti için 5 farklı model denedim. Öncelikle veri setimi %70 training ve %30 validation olarak ikiye ayrıdım. Training aşamalarında bazı modellerde cross-validation kullandım. Ayrıca Hyper parameter tuning methodları kullanarak hyper parametreleri seçtim.

4.1 Logistic Regresyon

Basit bir kullanıma sahip olan logistic regresyon ile öncelikle ilk binary classification modelimi denemiş oldum. Fakat çok iyi bir sonuç alamadım. Bu model yerine XGBOOST ve Random Forest gibi decision tree temelli classification problemleri denemeye karar verdim.

4.2 Multiplayer Perceptron

Multiplayer Porceptron kullanarak binary classification yapmak istedim. Elimdeki verinin az olmasından dolayı derin bir sinir ağı model kullanamadım. Fakat geniş ve sığ bir model kullanmak istedim. Bu model için ayrıca veri seti analizi yapmadan sadece oversampling yaparak deneme yaptım.

4.3 XGBOOST Classification

XGBOOST gradient tabanlı bir boosting karar ağacı modelidir. Boosting algoritmaları arasındaki fark genellikle zayıf öğrenicilerin eksiklerini öğrenmesi üzerinedir. Gradient Boosted Regresyon Ağaçları al-

goritması öncelikli olarak bir initial leaf oluşturulur. Sonrasında tahmin hataları göz önüne alınarak yeni ağaçlar oluşturulur. Bu durum karar verilen ağaç sayısına ya da modelden daha fazla gelişme kaydedilemeyinceye kadar devam eder. Yapılan bu tahminin ne kadar iyi olduğu modelin hatalı tahminleri ile incelenir.

4.4 CATBOOST Classification

CatBoost boş veriler ile başa çıkabilir, kategorik verilere encoding uygular. Kategorik veriler ile yüksek performanslı çalışabilmesinin nedeni, veri hazırlığı yaparken ayrıca bir kodlama işlemi yapılmasına gerek duyulmamasıdır. Veri'nin numerik hale getirilmemesi hem öğrenme hızını hem de sonuçların kalitesini etkilemektedir. Ayrıca Catboost simetrik ağaçlar kurar. Bu sayede çok derin ağaçlar kurmadan yüksek tahmin oranı yakalar ve overfitting problemini aşar.

Bu sebeple, modeli kullanırken 3.Bölümde bahsettiğim pre-processing işlemini uygulamadım.

4.5 Random Forest

Random Forest algoritması, birden çok karar ağacı üzerinden her bir karar ağacını farklı bir gözlem örneği üzerinde eğiterek çeşitli modeller üretip, sınıflandırma oluşturmanızı sağlamaktadır.

Algoritma veri seti üzerinde çeşitli modellerin oluşturulması ile setini yeniden ve daha derin şekilde eğitme imkanı sunmasıdır. Random forest modelinin diğer bir özelliği bize feature'ların ne kadar önemli olduğunu vermesi. Random forest algoritmasına x sayıda feature verip en faydalı y tanesini seçmesini isteyebiliriz ve istersek bu bilgiyi istediğimiz başka bir modelde kullanabiliriz.

5 Test Sonuçları

Projedeki başarım matriği olarak Accuracy kullandım. Hyper parametreleri fine tuning edebilmek için GridSearchCV kullandım.

5.1 Logistic Regresyon

Logistic Regresyon Modeli 0.71 accuracy ile 5 model arasında en düşük skoru aldı. Logistic regresyon modeli her ne kadar temel bir model bile olsa veri setindeki boyun fazla iken kötü sonuç veriyor.

5.2 Multiplayer Perceptron

Multiplayer Perceptron modeli ise 0.77 Accuracy ile daha iyi bir sonuç veriyor. Bu modeli eğitirken hyper parameter'ları şu şekilde belirlendi; Layerlar: 800, 400, 100 Activasyon fonksiyonu: Relu Optimization: Adam Learning Rate: 0.001 GridSearchCV fine tuning sonucunda ise bu değerleri aldım. Modelin daha iyi bir accuracy'e sahip olammasının veri setinin küçük olmasından kaynaklandığını düşünüyorum.

5.3 XGBOOST Classification

Bu modelde ise 0.81 ile yüksek bir accuracy skoruna ulaştım. Bu sayede hedefime ulaşmış oldum. Ensemble bir yöntem kullanan XGboost'u geliştirirken şu hyper parameter'larını kullandım. Layerlar: 800, 400, 100 Activasyon fonksiyonu: Relu Optimization: Adam Learning Rate: 0.01 max depth: 5 n estimators: 1000

5.4 CATBOOST Classification

CatBoost Modelini geliştirirken 3. kısımda belirttiğim pre processing adımlarını atladım. Cat boostun daha iyi performans vermesi için literatürde yapılan yöntemi izledim. Bu modelin test sonucunda ise 0.78 accuracy skoru aldım. Böylelikle diğer modellerden daha iyi bir sonuç elde etmiş oldum.

```
RESULTS :
CatBoost Model Accuracy :  0.78
CatBoost Model F1-score :  0.53
Classification Report :
              precision    recall  f1-score   support

         0.0       0.83      0.89      0.86      4278
         1.0       0.60      0.47      0.53      1470

    accuracy                           0.78      5748
   macro avg       0.71      0.68      0.69      5748
weighted avg       0.77      0.78      0.77      5748
```

Figure 4: CatBoost

5.5 Random Forest

Random forest modelinde ise 0.76 accuracy skoruna ulaştım. Bu modeli eğitirken hyper parameter'ları şu şekilde belirlendi; 'bootstrap': True, 'criterion': 'gini', 'max depth': 20, 'max leaf nodes': 20, 'n estimators': 100 Modelin daha iyi bir accuracy'e sahip olammasının veri setinin küçük olmasından kaynaklandığını düşünüyorum.

6 Sonuclar

Bu projede bir insan kaynakları veri seti üzerinde iş değiştirmek isteyen data scientist'leri sınıflandırmaya çalıştım. Veri seti üzerinde veri analizi, veri mühendisliği, makine öğrenmesi yöntemlerini uyguladım. Farklı makine öğrenmesi modelleri eğittik ve model üzerinde bir çıkarım yaptım.
Proje ile birlikte Makine Öğrenmesi dersinde öğrendiğim yöntemleri en çok deneyim kazanacak şekilde uygulamak istedim. Seçtiğim veri seti ile de bu hedefime ulaştım. Özellikle veri seti üzerine yapılan çalışmalardan faydalanarak üzerine kendi eklemlerimi yaptım ve derste öğrendiğim yöntemler üzerinde kafa yordum. Bu durum bana, model geliştirirken aslında ne kadar fazla parametreyi göz önünde bulundurmam gerektiği konusunda tecrübe kazandırdı.
Projede ilk hedefim olan %80 accuracy score'a ulaşmış oldum fakat hedeflediğim %85 accurac'ye ulaşamadım. Bunun sebebinin verinin sparse ve az olmasından kaynaklandığını düşünüyorum.
Gelecek çalışmalarda daha büyük ve dengeli veri setleri üzerinde insan kaynaklarının işini kolaylaştıracak modeller geliştirilebilir. Geliştirilen modeller sektörde kullanılabilir. Bu sayede bir şirketin en önemli departmanı olan insan kaynaklarının, zamanını bu tür sınıflandırma işlerine ayırmasının önüne geçilebilir. Yapay zeka modelleri kullanarak bu tür fazla vakit alan işler rahatlıkla çözüme kavuşabilir.

References

- Veri Seti
- Github Code Repository
- https://www.veribilimiokulu.com/catboost-nedir-diger-boosting-algoritmalarindan-farki-nelerdir/
- https://www.veribilimiokulu.com/xgboost-nasil-calisir/
- https://www.veribilimiokulu.com/siniflandirma-notlari-18-random-forest-python-uygulama/

```
RESULTS :
RandomForest Model Accuracy :  0.76
RandomForest Model F1-score :  0.54
Classification Report :
              precision    recall  f1-score   support

         0.0       0.85      0.83      0.84      2859
         1.0       0.53      0.55      0.54       973

    accuracy                           0.76      3832
   macro avg       0.69      0.69      0.69      3832
weighted avg       0.76      0.76      0.76      3832
```

Figure 5: Random Forest

```
RESULTS :
XGBoost Model Accuracy :  0.81
XGBoost Model F1-score :  0.5
Classification Report :
              precision    recall  f1-score   support

         0.0       0.83      0.90      0.87      2903
         1.0       0.59      0.44      0.50       929

    accuracy                           0.81      3832
   macro avg       0.71      0.67      0.69      3832
weighted avg       0.78      0.79      0.78      3832
```

Figure 6: XGBoost Classifier

	ROC-AUC	F1	precision_0	recall_0	precision_1	recall_1
LogisticRegression	0.714878	0.502905	0.829327	0.844701	0.517391	0.489209

Figure 7: Logistic Regresyon

```
Results on the test set:
              precision    recall  f1-score   support

         0.0       0.82      0.88      0.85      2903
         1.0       0.52      0.40      0.45       929

    accuracy                           0.77      3832
   macro avg       0.67      0.64      0.65      3832
weighted avg       0.75      0.77      0.75      3832
```

Figure 8: Multiplayer Perceptron