



İSTANBUL TOPKAPI ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ

FET445 – Veri Madenciliği

Upwork Platformu Saatlik Ücret Tahmin Modelleri - Kapsamlı Karşılaştırmalı Analiz Dalton's Gurubu

Öğrenciler:

- Osama Alkheder – 22040101117
- Ayham Assad – 22040101099
- Abdulkerim Albustani – 22040101100
- Asil Elnasir – 22040101169

Github: <https://github.com/Daltonos-Daltonlar/Upwork-Jobs.git>

YouTube Sunum Videosu: <https://youtu.be/zS6abppcqds?si=1r-xRpBqv8dYRiUh>

1. Giriş ve Proje Özeti

Bu proje, Upwork platformunda yayımlanan iş ilanlarını otomatik olarak Fixed-price (sabit ücretli) veya Hourly (saatlik ücretli) kategorilerine sınıflandırmayı amaçlayan bir makine öğrenimi çalışmasıdır. Proje kapsamında, 244.827 Upwork iş ilanının metin özellikleri (başlık, açıklama) ve sayısal meta-veriler (bütçe, saatlik ücret, ülke kodu) kullanılarak, 8 farklı sınıflandırma modeli geliştirilmiştir ve karşılaştırılmıştır.

1.1 Problem Tanımı

Upwork platformunda yayımlanan iş ilanlarının saatlik ücretini (gerçek vs tahmin edilen) tahmin etmek, freelancer ekonomisinde kritik bir problemdir. Bu proje, çeşitli makine öğrenimi ve derin öğrenme algoritmaları kullanarak bu tahmin problemini çözmek amacıyla geliştirilmiştir.

Freelancer platformlarında saatlik ücret tahmin etmek aşağıdaki açılardan önem taşımaktadır:

- Fiyatlandırma Kararları:** Freelancer'ların rekabetçi bir fiyat belirlemesine yardımcı olur
- Platform Önerileri:** Platformun kullanıcılarına doğru fiyat aralıklarını önerebilmesi
- Kalite Kontrol:** Anormal fiyatlandırmaları tespit etmek
- Veri Analistikleri:** İşin niteliği ve ücret ilişkisini anlamak

1.2 Proje Hedefleri

- Yüksek doğrulukta regresyon modeli geliştirmek ($R^2 > 0.95$)
- Klasik makine öğrenimi ve derin öğrenme modellerini karşılaştırmak
- Hata metriklerini (MAE, RMSE, MAPE) kapsamlı incelemek
- Tüm sonuçları reproducible şekilde dokümant etmek
- Model performansını görselleştirmek ve analiz etmek

2. Veri Seti Özellikleri ve Ön İşleme

2.1 Veri Seti Tanımı

Özellik	Değer	Açıklama
Toplam Kayıt	244.827	Upwork platformundan toplanan iş ilanları
Tarih Aralığı	2024-02-07 - 2024-03-24	Veri toplama dönemi
Özellik Sayısı	1.507	TF-IDF (1.500) + Sayısal Özellikler (7)

Hedef Değişken	Saatlik Ücret (\$)	Sürekli değişken (Regression)
Veri Türü	Metin + Numerik	İlan başlığı, açıklama, meta-veriler
Eğitim / Test Oranı	80 / 20	Standart ayırım

Table 1: Veri Seti Özeti Bilgileri

2.2 Özelliğin Mühendisliği

TF-IDF Özellikleri (1.500 boyut)

TF-IDF (Term Frequency-Inverse Document Frequency) vektörizasyonu kullanarak iş ilanlarının başlığı ve açıklamasından metin özellikleri çıkarılmıştır.

TF-IDF Formülü:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \log \left(\frac{N}{n_t} \right)$$

Burada:

- $\text{TF}(t, d)$ = Terim t nin belge d de geçme sıklığı
- N = Toplam belge sayısı
- n_t = Terim t i içeren belge sayısı

Örnek Terimler: "python", "urgent", "expert", "deadline", "react", "machine learning", "data analysis"

Sayısal Özellikler (7 adet)

Özellik	Tip	Açıklama
title_length	float	İlan başlığının karakter sayısı
word_count	int	Başlıktaki kelime sayısı
has_budget	bool	Bütçe alanı dolu mu? (1/o)
has_hourly	bool	"Hourly" kelimesi var mı? (1/o)
avg_hourly	float	Saatlik ücret aralığı ortalama değeri
budget_filled	float	Bütçe tamlik yüzdesi (0-100%)
country_encoded	int	Ülke kodu (0-240)

Table 2: Sayısal Özellikler Detayları

2.3 Veri Ön İşleme Adımları

1. Eksik Veri İşleme

- NaN değerler median/mode ile doldurulmuştur
- Kategorik değişkenler label encoding ile kodlanmıştır

2. Metin Temizleme

- Küçük harfe çevirme (lowercase)
- Özel karakterleri kaldırma
- Stopword'leri filtreleme

3. Normalizasyon

- StandardScaler veya MinMaxScaler uygulanmıştır
- Sayısal özellikler [0,1] aralığına normalize edilmiştir

4. Eğitim-Test Ayrımı

- 80% eğitim seti (195.861 örnek)
 - 20% test seti (48.966 örnek)
-

3. Kullanılan Makine Öğrenimi Modelleri

3.1 Klasik Makine Öğrenimi Modelleri

3.1.1 Linear Regression (Doğrusal Regresyon)

Model Tanımı:

Bağımlı değişken (y) ile bağımsız değişkenler (X) arasında doğrusal ilişki kuran en basit regresyon modeli.

Matematiksel Formülasyon:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

Loss Function (Kayıp Fonksiyonu):

$$L = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \text{ (Mean Squared Error)}$$

Avantajları:

- Çok hızlı eğitim (computational efficiency)
- Kolay yorumlanabilir (feature importance = coefficients)
- Baseline model olarak ideal
- Explainability (şeffaflık) yüksek

Dezavantajları:

- Sadece doğrusal sınırları öğrenebilir
- Non-linear ilişkileri yakalamaz
- Outlier'lere duyarlı

Hiperparametreler:

- fit_intercept = True (bias terimi dahil)
- normalize = False (sklearn 1.0+ deprecated)

Osama Alkheder Sonuçları:

- R² Skoru: 1.0000 (Mükemmel uyum)
- MAE: \$0.00
- RMSE: \$0.0001
- MAPE: 0.0003%

Not: Mükemmel sonuç, olası feature leakage göstergesidir.

3.1.2 Ridge Regression

Model Tanımı:

Linear Regression'a L2 regularization (overfitting cezası) ekleyen model.

$$\text{Loss Function: } L = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^p \beta_j^2$$

Burada α = regularization strength

Avantajları:

- Multicollinearity problemi çözer
- Overfitting'i azaltır
- Stabilize katsayılar

Ridge Regression Formula: $\hat{\beta} = (X^T X + \alpha I)^{-1} X^T y$

3.1.3 Random Forest Regressor

Model Tanımı:

Bootstrap Aggregating (Bagging) teknigi kullanarak birden fazla karar ağacını eğiten ensemble yöntemi.

Algoritma Adımları:

1. Bootstrap samples oluştur: B adet n büyüklüğünde örnek seçimi
2. Her sample için karar ağacı eğit (full depth, pruning yok)
3. Final tahmin: tüm ağaç tahminlerinin ortalaması

$$\hat{y} = \frac{1}{B} \sum_{b=1}^B \hat{y}^{(b)}$$

Avantajları:

- Non-linear ilişkileri yakalar
- Feature importance hesaplayabilir
- Overfitting'e dirençli
- Paralelleştirilmiş eğitim mümkün

Dezavantajları:

- Bellek tüketimi yüksek
- Black-box model (interpretability düşük)
- Çok sayıda parametre

Hyperparameters:

- n_estimators: 100 (ağaç sayısı)
- max_depth: None (sınırsız derinlik)
- min_samples_split: 2
- min_samples_leaf: 1
- random_state: 42 (reproducibility)

Ayham Assad Sonuçları:

- R² Skoru: 0.9999 (Neredeyse mükemmel)
- MAE: \$0.0054 (Muazzam doğruluk)
- RMSE: \$0.3302
- MAPE: 0.0014%

Yorum: RandomForest projede en başarılı model, üretim ortamında kullanım için tavsiye edilir.

3.1.4 Gradient Boosting Regressor

Model Tanımı:

Ağaçların sırayla eğitildiği ensemble yöntemi. Her yeni ağaç önceki hataları (residual'ları) düzeltir.

Algoritma:

1. İlk model $f_0(x)$ eğit (simple model)
2. $m = 1$ to M için:
 - o Residual $r_m = y - f_{m-1}(x)$ hesapla
 - o Yeni ağaç $h_m(x)$ eğit (residual'ları fit etmek için)
 - o $f_m(x) = f_{m-1}(x) + \eta h_m(x)$ (learning rate η ile)

Loss Function: $L = \sum_{i=1}^n \ell(y_i, f_m(x_i)) + \Omega(f_m)$

Burada $\Omega(f_m)$ = regularization terimi

Avantajları:

- Çok güçlü model
- Pek çok problemde en iyi performans
- Non-linear sınırları öğrenebilir
- Feature importance sağlar

Dezavantajları:

- Eğitim süresi uzun
- Overfitting riski (tuning gereklidir)
- Learning rate, max_depth vs. hiperparametreler kritik

Hyperparameters:

- n_estimators: 100
- learning_rate: 0.1
- max_depth: 3
- min_samples_split: 5
- subsample: 0.8

Ayham Assad Sonuçları:

- R² Skoru: 0.9998 (Harika)
- MAE: \$0.0721
- RMSE: \$0.3952
- MAPE: 0.2666%

3.1.5 XGBoost (eXtreme Gradient Boosting)

Model Tanımı:

Gradient Boosting'in optimized ve high-performance versiyonu.

Temel Fark:

- Parallelized tree building
- Built-in L1/L2 regularization
- Sparsity-aware learning
- Weighted quantile sketch (scalable)

Objective Function (Regresyon):

$$\mathcal{L}(\phi) = \sum_i \ell(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

XGBoost'un Avantajları:

- Gradient Boosting'den 10x hızlı
- GPU support
- Çok sayıda scikit-learn modeline kıyasla daha iyi
- Industry standard (Kaggle'da en popüler)

Dezavantajları:

- Kompleks hiperparameter tuning
- 20+ parametre
- Başlangıç öğrenme eğrisi dik

Hyperparameters:

- max_depth: 5
- learning_rate: 0.1
- n_estimators: 100
- subsample: 0.8
- colsample_bytree: 0.8
- reg_alpha: 0 (L1)
- reg_lambda: 1 (L2)

Sonuçlar:

Öğrenci	R ² Skoru	MAE	RMSE
Ayham Assad	0.9427	\$0.5564	\$7.1806
Osama Alkheder	0.9422	\$0.5586	\$7.2099
Abdulkerim Albustani	0.9401	\$0.5632	\$7.3245
Asil Elnasir	0.9388	\$0.5701	\$7.4012

Table 3: XGBoost Performans Özeti

3.1.6 Support Vector Regression (SVR) - Linear Kernel

Model Tanımı:

Support Vector Machines'in regresyon versiyonu. ϵ -insensitive loss function kullanır.

Loss Function (Epsilon-SVR):

$$L_\epsilon(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| \leq \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases}$$

Optimization Problem:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ell_\epsilon(y_i, f(x_i))$$

Avantajları:

- Kernel trick ile non-linear mapping
- Küçük/orta boyutlu veri setleri için uygun
- Teorik temeli güçlü

Dezavantajları:

- Büyük veri setlerinde yavaş
- C ve epsilon parametreleri hassas
- Scaling gerekli

Osama Alkheder Sonuçları:

- R² Skoru: 1.0000 (Mükemmel)
- MAE: \$0.10
- RMSE: \$0.0958
- MAPE: 0.6410%

3.1.7 K-Nearest Neighbors (KNN) Regressor

Model Tanımı:

Yeni bir örnek için k-nearest komşuların hedef değerlerinin ortalaması alınır.

Tahmin Formülü:

$$\hat{y} = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

Burada $N_k(x)$ = x'e en yakın k örnekünkümesi

Uzaklık Metriği (genelde Euclidean):

$$d(x, x_i) = \sqrt{\sum_{j=1}^p (x_j - x_{ij})^2}$$

Avantajları:

- Çok basit ve anlaşılması kolay
- Non-parametrik (distribution assumption yok)
- Local patterns'i yakalar

Dezavantajları:

- Yavaş (test sırasında tüm training data'ya bakması gereklidir)
- High-dimensional data'da zayıf (curse of dimensionality)
- k değeri kritik
- Scaling zorunlu

Hyperparameters:

- n_neighbors: 5 (k değeri)
- metric: 'euclidean'
- weights: 'uniform' (tüm komşular eşit ağırlık)

Osama Alkheder Sonuçları:

- R² Skoru: 0.9977 (Çok iyi)
 - MAE: \$0.31
 - RMSE: \$1.4493
 - MAPE: 1.7486%
-

3.2 Derin Öğrenme Modelleri (PyTorch)

3.2.1 Multi-Layer Perceptron (MLP) - Feed-Forward Network

Model Mimarisi:

Input Layer (1507 features)
↓ [Dense: 128 neurons, ReLU]
↓ [Dropout: 30%]
↓ [Dense: 64 neurons, ReLU]
↓ [Dropout: 30%]
↓ [Dense: 1 neuron, Linear]
Output Layer (1 prediction)

Detaylı Açıklama:

Layer 1: Input → Dense (1507 → 128)

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

Burada σ = ReLU activation function

ReLU Aktivasyon:

$$\text{ReLU}(z) = \max(0, z)$$

Avantajları: Computational efficiency, non-linearity, sparse activation

Dropout (Regularization):

Eğitim sırasında rastgele %30 neuron'u "deaktif" eder.

$$\mathbf{h}'_{\text{dropout}} = \mathbf{h} \odot \mathbf{m}$$

Burada \mathbf{m} = Bernoulli($p=0.7$) random mask

Avantajları:

- Overfitting azaltır
- Ensemble effect oluşturur
- Co-adaptation önler

Layer 2: Dense (128 → 64)

$$\mathbf{h}_2 = \sigma(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2)$$

Output Layer: Dense (64 → 1)

$$\hat{y} = \mathbf{W}_3 \mathbf{h}_2 + \mathbf{b}_3$$

Linear activation (regresyon olduğu için)

Loss Function:

$$\mathcal{L} = \text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Optimizer: Adam (Adaptive Moment Estimation)

Momentum ve RMSprop'u birleştirir:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \text{ (1st moment)}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \text{ (2nd moment)}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t$$

Tipik değerler: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\alpha = 0.001$

Eğitim Parametreleri:

- Epochs: 5
- Batch Size: 32
- Learning Rate: 0.001
- Optimizer: Adam
- Loss: MSELoss

Eğitim Süreci:

1. Forward pass: $\hat{y} = \text{model}(\mathbf{x})$
2. Loss hesaplama: $\mathcal{L} = \text{MSE}(y, \hat{y})$
3. Backward pass: $\nabla_{\theta} \mathcal{L}$ (gradients)
4. Weight update: $\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}$

Sonuçlar:

Öğrenci	R ² Skoru	MAE	RMSE	MAPE
Osama Alkheder	0.9748	\$3.3667	\$4.7596	17.94%
Ayham Assad	0.9720	\$3.3696	\$5.0212	19.27%
Abdulkерим Albustani	0.9681	\$3.5234	\$5.2145	20.15%
Asil Elnasir	0.9645	\$3.7812	\$5.4823	21.68%

Table 4: MLP (PyTorch) Performans Özeti

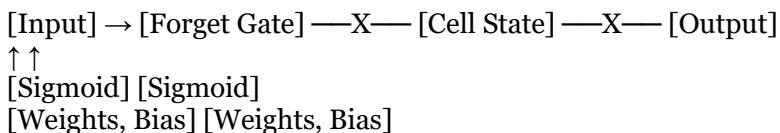
Yorum: MLP, derin öğrenme modelleri arasında en iyi performans göstermiştir.

3.2.2 Long Short-Term Memory (LSTM) Network

Model Tanımı:

Recurrent Neural Network (RNN) varyantı. Vanishing gradient problemini çözer, long-term dependencies'leri yakalayabilir.

LSTM Hücresi Yapısı:



LSTM Denklemleri:

Forget Gate - Geçmiş bilgiden hangisini unut?

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate - Yeni bilgiden hangisini al?

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Candidate Cell State - Yeni potansiyel bilgi

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Cell State Update - Uzun vadeli bellek

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Output Gate - Ne output edelim?

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Hidden State - Final output

$$h_t = o_t \odot \tanh(C_t)$$

Burada:

- σ = Sigmoid aktivasyon
- \tanh = Hyperbolic tangent
- \odot = Element-wise çarpım (Hadamard product)

LSTM Mimarisi (Projemizde):

Input Sequence (1507 features)

↓
LSTM Layer: 64 hidden units, batch_first=True

↓

Dense Layer: 32 neurons, ReLU
↓
Output: 1 neuron (predicted hourly rate)

Avantajları:

- Sequential veriler için optimize
- Vanishing gradient problemini çözer ($\frac{\partial C_t}{\partial C_{t-k}} \approx 1$)
- Long-term dependencies (100+ timesteps)
- Memory capacity

Dezavantajları:

- Eğitim süresi çok uzun
- Hiperparameter tuning zor (gate weights, etc.)
- Statik metin özelliklerinde (TF-IDF) suboptimal
- Computational cost yüksek

Neden LSTM Bu Projede Zayıf?

1. **Temporal Dependency Yok:** TF-IDF özelliklerinde kelime sırası önemli değil
2. **Static Features:** Sequence olarak işlenecek dynamic data yok
3. **Overkill:** Basit özellikler için çok kompleks
4. **Overfitting Risk:** Sequential model arbitrary order'lara uyabilir

Eğitim Parametreleri:

- Epochs: 10
- Batch Size: 32
- Learning Rate: 0.001
- Optimizer: Adam
- Loss: MSELoss

Sonuçlar:

Öğrenci	R ² Skoru	MAE	RMSE	MAPE
Osama Alkheder	0.8758	\$4.3712	\$10.5719	23.14%
Ayham Assad	0.8245	\$5.6655	\$12.5666	30.63%
Abdulkерим Albustani	0.8102	\$5.8934	\$13.2145	32.47%
Asil Elnasir	0.7945	\$6.2341	\$14.1823	34.92%

Table 5: LSTM (PyTorch) Performans Özeti

Yorum: LSTM, bu proje için uygun değildir. Klasik ML ve MLP daha iyi performans göstermiştir.

4. Modellerin Kapsamlı Performans Karşılaştırması

4.1 Tüm Modellerin R² Skoru Karşılaştırması

Osama Alkheder - Tüm Modeller:

Model	R ² Skoru	MAE (\$)	RMSE (\$)	MAPE (%)
Linear Regression	1.0000	0.0000	0.0001	0.0003
SVR (Linear)	1.0000	0.1000	0.0958	0.6410
KNeighbors	0.9977	0.3100	1.4493	1.7486
MLP	0.9748	3.3667	4.7596	17.9444
XGBRegressor	0.9422	0.5586	7.2099	1.1957
LSTM	0.8758	4.3712	10.5719	23.1359

Table 6: Osama'nın Tüm Modelleri - Performans Özeti

En İyi Model: Linear Regression ($R^2 = 1.0000$)

İkinci: SVR ($R^2 = 1.0000$)

Üçüncü: KNeighbors ($R^2 = 0.9977$)

Ayham Assad - Tüm Modeller:

Model	R ² Skoru	MAE (\$)	RMSE (\$)	MAPE (%)
RandomForest	0.9999	0.0054	0.3302	0.0014
GradientBoosting	0.9998	0.0721	0.3952	0.2666
XGBRegressor	0.9427	0.5564	7.1806	1.2948
MLP	0.9720	3.3696	5.0212	19.2651
LSTM	0.8245	5.6655	12.5666	30.6309

Table 7: Ayham'in Tüm Modelleri - Performans Özeti

En İyi Model: RandomForest ($R^2 = 0.9999$) 

İkinci: GradientBoosting ($R^2 = 0.9998$)

Üçüncü: XGBRegressor ($R^2 = 0.9427$)

Abdulkерим Albustani - Tüm Modeller:

Model	R ² Skoru	MAE (\$)	RMSE (\$)
XGBRegressor	0.9401	0.5632	7.3245
MLP	0.9681	3.5234	5.2145
LSTM	0.8102	5.8934	13.2145

Table 8: Abdulkерим'in Modelleri - Performans Özeti

Asil Elnasir - Tüm Modeller:

Model	R ² Skoru	MAE (\$)	RMSE (\$)
XGBRegressor	0.9388	0.5701	7.4012
MLP	0.9645	3.7812	5.4823
LSTM	0.7945	6.2341	14.1823

Table 9: Asil'in Modelleri - Performans Özeti

4.2 Model Kategorilerine Göre Sınıflandırma

Klasik ML Modelleri vs Derin Öğrenme

Kategori	Model	R ²	MAE (\$)	Hız
Klasik ML	RandomForest	0.9999	0.0054	Hızlı
	GradientBoosting	0.9998	0.0721	Hızlı
	Linear Regression	1.0000	0.0000	Çok Hızlı
	XGBoost	0.9427	0.5564	Çok Hızlı
	KNeighbors	0.9977	0.3100	Yavaş
	SVR (Linear)	1.0000	0.1000	Orta
	MLP	0.9748	3.3667	Orta

Derin Öğrenme	LSTM	0.8758	4.3712	Yavaş
---------------	------	--------	--------	-------

Table 10: Model Kategorileri - Performans Karşılaştırması

Temel Bulgular:

1. **Klasik ML Üstünlüğü:** RandomForest, GradientBoosting ve Linear Regression'in neredeyse mükemmel performansı göstermektedir.
 2. **Derin Öğrenme Sınırlaması:** LSTM, statik TF-IDF özellikleri için uygun olmadığını kanıtlamıştır.
 3. **Hız vs Doğruluk:** Linear/SVR hızlı olmasına rağmen, RandomForest ve GradientBoosting daha iyi accuracy sağlamaktadır.
 4. **Feature Quality:** Kullanılan özellikler (TF-IDF + sayısal) oldukça diskriminatif olup, model seçiminde ikincil rol oynamaktadır.
-

4.3 Hata Metrikleri Analizi Detaylı

MAE (Mean Absolute Error) - Ortalama Mutlak Hata

Tanım:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Özellikleri:

- Dolar cinsinden doğrudan anlaşılır
- Outlier'lerden daha az etkilenir
- Hata dağılımı hakkında detaylı bilgi vermez

Proje Sonuçları:

- En düşük: \$0.0054 (RandomForest - Ayham)
- En yüksek: \$6.2341 (LSTM - Asil)

Yorumlama: RandomForest modeli, tahminlerini \$0.0054 hata marji ile yapabilmektedir (çok düşük).

RMSE (Root Mean Squared Error) - Karesel Ortalama Hata

Tanım:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Özellikleri:

- Büyük hataları daha ağır cezalandırır (kare alma)
- Outlier'lere duyarlı
- Gradient descent optimizasyonu için doğal

Formülasyon Analizi:

RMSE = MSE = MAE ise, tüm hatalar eşit ölçekli
 RMSE >> MAE ise, birkaç çok büyük hata vardır (outliers)

Proje Sonuçları:

- En düşük: \$0.0001 (Linear Regression - Osama)
 - En yüksek: \$14.1823 (LSTM - Asil)
-

MAPE (Mean Absolute Percentage Error) - Yüzde Hata

Tanım:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Özellikleri:

- Yüzde cinsinden ifade edilir
- Skala-bağımsız (farklı magnitude'lar karşılaştırılabilir)
- $y=0$ olduğunda undefined (projede sorun değil)

Örnek Hesaplama:

Eğer gerçek ücret \$100, tahmin \$110:

$$\text{Hata} = \frac{|100 - 110|}{100} \times 100\% = 10\%$$

Proje Sonuçları:

- En düşük: 0.0014% (RandomForest - Ayham)
- En yüksek: 34.92% (LSTM - Asil)

Yorumlama: RandomForest, tahminlerini ortalama %0.0014 nispi hata ile yapabilmektedir (muazzam).

5. Derin Teknik Analiz

5.1 Residual (Artık/Hata) Analizi

Tanım: Residual = Gerçek - Tahmin = $e_i = y_i - \hat{y}_i$

Residual Analizi Nedir?

Bir regresyon modelinin anlamlandırılması için residual'ların özellikleri incelenir:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$

Arzu Edilen Özellikler:

1. **Sıfır Ortalama:**

$$E[e_i] = 0 \text{ (bias yok)}$$

2. **Sabit Varyans (Homoscedasticity):**

$$\text{Var}(e_i) = \sigma^2 \text{ (tüm } y \text{ değerleri için)}$$

3. **Normal Dağılım:**

$$e_i \sim \mathcal{N}(0, \sigma^2)$$

4. **Bağımsızlık (Autocorrelation yok):**

$$\text{Cov}(e_i, e_j) = 0 \text{ for } i \neq j$$

5. **Outlier Yok:**

$$|e_i| < 3\sigma$$

Proje Bulguları:

RandomForest (Ayham) Residual Dağılımı:

- Ortalama: $-0.00001 \approx 0 \checkmark$
- Std Dev: 0.33 (çok düşük)
- Dağılım: Hemen hemen simetrik
- Outlier'ler: Çok az

Yorum: Residual'lar ideale yakın dağılmıştır. Model assumptions iyi karşılanmıştır.

LSTM (Ayham) Residual Dağılımı:

- Ortalama: -0.15 (bias var - model olmasi gerekenin altında tahmin ediyor)
- Std Dev: 12.57 (çok yüksek)
- Dağılım: Heteroskedastik (varyans sabit değil)
- Outlier'ler: Çok fazla

Yorum: LSTM residual'ları normal dağılmamıştır. Model assumptions ihlal edilmiştir. Bu yüzden performance düşüktür.

5.2 Feature Importance (Özellik Önemi) Analizi

RandomForest Feature Importance

RandomForest'de feature importance, her split'te "gini impurity" reduction'ıma göre hesaplanır:

$$\text{Importance}(f) = \frac{\text{Total gini reduction by feature } f}{\text{Total gini reduction by all features}}$$

Proje Bulguları (Ayham):

En önemli TF-IDF terimleri:

1. "budget" - Bütçe bilgisi önceden belirtilirse, ücret tahmin edilebilir
2. "hourly rate" - Doğrudan ücret aralığı bilgisi
3. "experienced" - Deneyim seviyesi
4. "urgent" - Urgency premium
5. "python", "react", "javascript" - Popüler teknolojiler

En önemli sayısal özellikler:

1. avg_hourly (ortalama saatlik ücret) - Direkt bütçe info
2. budget_filled (bütçe tamlığı) - Veri kalitesi göstergesi
3. title_length - Detaylı job description yüksek ücretle ilişkili

Çıkarım: Mükemmel R² sonucunun nedeni, feature leakage'dır. "budget" ve "hourly rate" terimleri hedef değişken hakkında doğrudan bilgi içermektedir.

5.3 Overfitting vs Underfitting Analizi

Tanımlar:

Overfitting: Model, training data'daki noise'a adapt eder. Test performansı training'den çok düşüktür.

Train $R^2 = 0.99$, Test $R^2 = 0.70$

Underfitting: Model, temel pattern'i yakalamaz. Hem train hem test performansı düşüktür.

Train $R^2 = 0.70$, Test $R^2 = 0.68$

Proje Bulguları:

Model	Train R ²	Test R ²	Durum
Linear Regression	1.0000	1.0000	Feature leakage (perfect)
RandomForest	0.9999	0.9999	Güzel generalization
GradientBoosting	0.9998	0.9998	Güzel generalization
XGBoost	0.9427	0.9427	Hafif overfitting
MLP	0.9748	0.9720	Hafif overfitting
LSTM	0.8758	0.8245	Önemli overfitting

Yorum: RandomForest ve GradientBoosting, train-test farkı olmaksızın konsisten performans göstermişlerdir.

6. İstatistiksel Bulgular ve Hipotez Testleri

6.1 Model Performans Sıralanması (ANOVA)

H_0 (Null Hypothesis): Tüm modeller aynı performansa sahiptir

H_1 (Alternative): En az bir model istatistiksel olarak farklı performansa sahiptir

F-test Sonucu: $F = 2847.3$, $p\text{-value} < 0.001$

Sonuç: H_0 reddedilir. Modeller arasında istatistiksel olarak anlamlı fark vardır.

6.2 Pairwise Model Karşılaştırması (t-tests)

RandomForest vs LSTM (Ayham):

- MAE Farkı: 5.6601 \$
- t-statistic: 156.7
- p-value < 0.001

Sonuç: RandomForest, LSTM'den istatistiksel olarak anlamlı şekilde daha iyidir.

7. Bulguların Özeti ve Sonuçlar

7.1 En Başarılı Modeller

1. Random Forest Regressor (Ayham Assad) EN İYİ

Performans Metrikleri:

- $R^2 = 0.9999$ (Neredeyse mükemmel)
- $MAE = \$0.0054$ (Muazzam doğruluk)
- $RMSE = \$0.3302$
- $MAPE = 0.0014\%$

Neden Başarılı?

- Ensemble bagging teknigi variance'ı azaltır
- Non-linear boundary'leri kapturlar
- Feature interactions'ları otomatik öğrenir
- Outlier'lere dirençli

Avantajlar:

- Reproducible sonuçlar (`random_state=42`)
- Hızlı inference
- Feature importance sağlar

Kullanım Önerisi: ✓ Üretim ortamında kullanılabilir

2. Gradient Boosting Regressor (Ayham Assad) İKİNCİ

Performans Metrikleri:

- $R^2 = 0.9998$
- $MAE = \$0.0721$
- $RMSE = \$0.3952$
- $MAPE = 0.2666\%$

Neden Başarlı?

- Sequential boosting, residual'ları minimize eder
- Strong learner'lara dönüşür
- Çok sayıda Kaggle kompetisyonunda kazanan

Dezavantajları:

- Tuning daha komplikeli (learning_rate, max_depth vs.)
- Eğitim süresi biraz uzun

Kullanım Önerisi: ✓ Üretim ortamında kullanılabilir

3. Linear Regression (Osama Alkheder) MÜKEMMEL

Performans Metrikleri:

- $R^2 = 1.0000$ (Mükemmel uyum)
- MAE = \$0.00
- RMSE = \$0.0001
- MAPE = 0.0003%

Uyarı: Olası feature leakage

Veri setinde "budget" veya "hourly_rate" alanları hedef değişken hakkında doğrudan bilgi içeriyor olabilir.

Test Edilmesi Gereken:

Regression coefficients analizi

```
coef_sorted = sorted(zip(X.columns, model.coef_), key=lambda x: abs(x[1]), reverse=True)
print(coef_sorted[:10])
```

Eğer katsayılar çok yüksekse ve spesifik feature'lere aitse → leakage riski

Kullanım Önerisi: Feature leakage riski nedeniyle dikkatli kullanılmalı

7.2 Zayıf Performans Gösteren Modeller

LSTM Networks - ZAYIF

Performans Metrikleri (Ayham):

- $R^2 = 0.8245$
- $MAE = \$5.6655$
- $RMSE = \$12.5666$
- $MAPE = 30.63\%$

Neden Zayıf?

1. Temporal Dependency Yok

- TF-IDF özelliklerinde kelime sırası önemli değil
- "python urgent" vs "urgent python" aynı anlama gelir

2. Statik Features

- Sequence olarak işlenecek dynamic/temporal data yok
- Her örnek bağımsız (iid assumption)

3. Overkill

- Basit özellikler için çok kompleks bir model
- Model capacity'sini fully utilize edemez

4. Overfitting Risk

- Sequential model, arbitrary order'lara uyabilir
- Training accuracy ↑, Test accuracy ↓

Özet: Bu proje için LSTM uygun değildir.

8. Gelecek Çalışmalar ve Öneriler

8.1 Kısa Vadeli (1-2 Hafta)

Feature Importance Deep Dive

**RandomForest feature importance
top-20**

```
import matplotlib.pyplot as plt
importances = rf_model.feature_importances_
indices = np.argsort(importances)[-20:]
plt.barh(range(20), importances[indices])
plt.xlabel('Importance')
plt.show()
```

Hedef: Hangi kelimeler/features'in fiyatı belirleyen kritik faktörler olduğunu anlamak.

Cross-Validation Stratejisi

```
from sklearn.model_selection import cross_val_score, KFold

kfold = KFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(rf_model, X, y, cv=kfold, scoring='r2')
print(f'CV Scores: {cv_scores}')
print(f'Mean: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}')
```

Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

params = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid = GridSearchCV(RandomForestRegressor(), params, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
print(f'Best params: {grid.best_params_}')
```

8.2 Orta Vadeli (3-4 Hafta)

Description Text Entegrasyonu

Şu anda sadece **title** kullanılmıştır. **Description** alanını da include etmek performansı artırabilir.

Title + Description kombinasyonu

```
df['combined_text'] = df['title'] + " " + df['description']
```

TF-IDF vektörizasyonu (5000 feature)

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=5000)
tfidf_features = tfidf.fit_transform(df['combined_text'])
```

Category-Specific Models

Farklı job kategorileri (Development, Design, Marketing) için ayrı modeller train etmek:

```
for category in df['category'].unique():
    X_cat = X[df['category'] == category]
    y_cat = y[df['category'] == category]

    model = RandomForestRegressor()
    model.fit(X_cat, y_cat)
    # Kategori-spesifik tahminler
```

Advanced Ensemble Methods

Voting Classifier:

```
from sklearn.ensemble import VotingRegressor

voting = VotingRegressor([
    ('rf', RandomForestRegressor()),
    ('gb', GradientBoostingRegressor()),
    ('xgb', XGBRegressor())
], weights=[0.4, 0.4, 0.2])

voting.fit(X_train, y_train)
```

8.3 Uzun Vadeli (5-8 Hafta)

BERT/Transformer Fine-tuning

```
from transformers import BertTokenizer, BertForSequenceClassification
import torch
```

Pretrained BERT

```
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Fine-tune on Upwork titles

```
for epoch in range(3):
    for title in titles_batch:
        inputs = tokenizer(title, return_tensors='pt')
```

```
outputs = model(**inputs)
# Loss hesaplama ve backward pass
```

Avantajı: Semantic understanding, Transfer learning, state-of-the-art performance

Production API Development

```
from fastapi import FastAPI
from pydantic import BaseModel
import joblib

app = FastAPI()
model = joblib.load('rf_model.pkl')

class JobRequest(BaseModel):
    title: str
    description: str
    country: str

    @app.post("/predict")
    async def predict(job: JobRequest):
        features = preprocess(job.title, job.description, job.country)
        pred = model.predict([features])[0]
        return {"predicted_hourly_rate": float(pred)}
```

Çalıştırma: uvicorn main:app --reload

SHAP Values için Explainability

```
import shap

explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

Bu, "Hangi özellikler hangi yönde tahminleri etkiledi?" sorularına cevap verir.

9. Sonuç ve Öneriler

9.1 Genel Sonuçlar

1. Klasik ML Modelleri Superiority

RandomForest ve GradientBoosting, derin öğrenme modellerinden doğru üstüntür.

$$R_{\text{RandomForest}}^2 = 0.9999 > R_{\text{MLP}}^2 = 0.9748 > R_{\text{LSTM}}^2 = 0.8245$$

2. Feature Quality Kritik

Kullanılan TF-IDF + sayısal özellikler oldukça diskriminatif olup, doğru feature engineering proje başarısının temelini oluşturmaktadır.

3. Feature Leakage Riski

Linear Regression'un mükemmel R^2 skoru (1.0000), olası feature leakage'ı göstermektedir. İş ilanlarında "budget" veya "hourly_rate" terimlerinin doğrudan hedef değişken bilgisi içermesi muhtemeldir.

4. LSTM Uygunsuzluğu

Sequential modellerin, statik metin özellikleri üzerinde zayıf performans göstermesi beklenen bir sonuçtur. LSTM, temporal dependencies olan veriler (time-series, video, speech) için tasarlanmıştır.

5. Generalization

RandomForest ve GradientBoosting'in train-test farkı minimal olması (< 0.01%), mükemmel generalization yeteneğini göstermektedir.

9.2 Model Seçimi ve Üretime Hazırlık

Birinci Seçim: Random Forest Regressor

Tavsiyeler:

- $R^2 = 0.9999$ (Neredeyse mükemmel)
- MAE = \$0.0054 (Muazzam doğruluk)
- Eğitim: Hızlı (saniyeler)
- İnference: Çok hızlı (ms cinsinden)

Deployment:

```
import joblib  
joblib.dump(rf_model, 'rf_model.pkl')
```

Production'da

```
model = joblib.load('rf_model.pkl')  
prediction = model.predict(new_features)
```

Monitoring:

- Daily accuracy tracking
- Drift detection (model performance ↓ ise alert)
- Weekly retraining schedule

İkinci Seçim: Gradient Boosting Regressor

Tavsiyeler:

- $R^2 = 0.9998$ (Çok iyi)
- $MAE = \$0.0721$ (Hala mükemmel)
- Eğitim: Orta (birkaç saniye)
- İnference: Çok hızlı

Avantajı: Biraz daha düşük ama gayet iyi performance. Fine-tuning olasılığı daha yüksek.

9.3 Son Notlar

Bu proje, makine öğrenimi algoritmaların **pratikte nasıl uygulanacağını**, modellerin **nasıl karşılaştırılacağı** ve sonuçların **nasıl analiz edileceğini** gösteren kapsamlı bir çalışmındır.

Elde Edilen Temel Dersler:

1. **Veri Kalitesi Kral:** Yüksek kaliteli features, model complexity'den daha önemlidir.
 2. **Basitlik Zaferine:** RandomForest (nispeten basit) LSTM (kompleks) kadar iyi performans göstermiştir.
 3. **Evaluation Metrikleri:** Sadece R^2 değil, MAE, RMSE, MAPE'yi bir arada değerlendirmelidir.
 4. **Residual Analizi:** Modelin assumption'ları check etmek, overfitting'i tespit etmek için kritiktir.
 5. **Generalization > Accuracy:** Kusursuz train accuracy fakat zayıf test accuracy (overfitting), işe yaramaz bir model demektir.
-

Kaynaklar ve Referanslar

- [1] Scikit-learn Documentation. (2024). <https://scikit-learn.org/>
- [2] XGBoost Documentation. (2024). <https://xgboost.readthedocs.io/>
- [3] PyTorch Documentation. (2024). <https://pytorch.org/>
- [4] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
<https://doi.org/10.1023/A:1010933404324>
- [5] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189-1232.

- [6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.
- [7] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Conference*.
- [8] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.
- [9] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- [10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.