**Controlix**

**Veri Bilimi Çalışması**

**1) Proje Konusunu Seçme**

1. **Alan Seçimi (Domain)**
   * İlgi duyduğunuz veya şirket için anlamlı olabilecek **herhangi bir alanı** seçebilirsiniz (ör. e-ticaret, finans, sağlık, sosyal medya, metin analizi, IoT sensör verileri vb.).
   * Eğer gerçek veri yoksa, **Kaggle** veya benzeri veri platformlarından (UCI Machine Learning Repository, Google Dataset Search vb.) domain’e uygun bir veri seti seçilebilir.
2. **Problem Türü**
   * **Tahmin (Regression)** veya **Sınıflandırma (Classification)** ya da **kümeleme (Clustering)** problemi olabilir.
   * Örnek:
     + Regresyon: Ev fiyat tahmini, satış tahmini, talep tahmini.
     + Sınıflandırma: Müşteri churn tahmini, sahtekarlık (fraud) tespiti, hasta teşhisi vb.
     + Kümeleme: Müşteri segmentasyonu, ürün kategorilendirme, benzer belgelerin gruplandırılması.

**Görev**: Stajyerin bu aşamada regresyon, sınıflandırma ve kümeleme için net bir **problem tanımı** yapması, hangi değişken(ler)i tahmin edeceğini veya neyi ortaya çıkarmak istediğini belirlemesi gerekiyor.

**2) Veri Toplama ve Keşif (Data Gathering & Exploration)**

1. **Veri Kaynaklarını Belirleme**
   * Seçilen veri seti hangi kaynaktan geliyor? (Kaggle, şirket veritabanı, API, CSV dosyası vb.)
   * Verinin boyutu, formatı, veri sözlüğü (metadata) gibi bilgiler kayda geçsin.
2. **Verinin İlk İncelemesi (Exploratory Data Analysis - EDA)**
   * Veri setindeki sütunlar (features) nelerdir, hedef (target) sütunu hangisidir?
   * Özet istatistikler: ortalama, standart sapma, medyan, min-max değerleri, vb.
   * Veri türleri (sayısal, kategorik, metin, zaman serisi vb.) ve dağılımlar.
3. **Veri Görselleştirme**
   * Dağılım grafikleri (histogram, boxplot) ile veri dağılımlarının incelenmesi.
   * Sayısal değişkenler için korelasyon analizleri ve ısı haritası (correlation matrix, heatmap).
   * Kategorik değişkenlerin bar chart/pie chart ile incelenmesi.
   * Zaman serisi söz konusuysa trend analizleri (line plot) vb.

**Görev**: Jupyter Notebook veya benzeri bir ortamda (R, Python, vs.) temel EDA çıktılarını sunmak ve veri setini anlatan bir rapor ( markdown hücreleriyle veya PDF ) oluşturmak.

**3) Veri Temizleme ve Özellik Mühendisliği (Data Cleaning & Feature Engineering)**

1. **Veri Temizleme (Cleaning)**
   * Eksik verileri (missing values) tespit etme: Silme, ortalama/medyan/bir mod değeri ile doldurma veya gelişmiş yöntemler (MICE, KNN imputer) kullanma.
   * Aykırı değerleri (outliers) tespit ve gerekiyorsa temizleme (ör. IQR yöntem, z-score vb.).
   * Tutarsız veya hatalı kayıtların incelenip düzeltilmesi ya da veri setinden çıkarılması.
2. **Özellik Seçimi ve Dönüşümleri (Feature Engineering)**
   * Kategorik verilerin dönüştürülmesi: One-hot encoding, label encoding, target encoding vb.
   * Sayısal verilerin standartlaştırılması (StandardScaler) veya normalleştirilmesi (MinMaxScaler).
   * Tarih/saat verisinden ek özellikler türetme (yıl, ay, gün, hafta sonu, tatil bilgisi vb.)
   * Metin verisinden n-gram, TF-IDF veya word embedding (örn. Word2Vec) çıkarmak.
   * Gerekiyorsa boyut indirgeme (PCA, t-SNE) çalışmaları.
   * Özellik kombinasyonları oluşturma (örn. etkileşim terimleri: feature1 \* feature2 vb.).
3. **Özellik Seçimi / Boyut Azaltma**
   * Yüksek boyutlu veride (çok sayıda sütun) önemli olmayan veya çok düşük varyansa sahip sütunları atma.
   * Korelasyonu çok yüksek olan sütunları (multicollinearity) temizleme veya birleştirme.

**Görev**: Tüm bu adımları, hangi kararların neden alındığını açıklayan **kısa bir dokümantasyon** şeklinde hazırlamak.

**4) Modelleme (Model Construction)**

1. **Doğrusal/Sade Modeller**
   * Basit bir **Lineer Regresyon** veya **Lojistik Regresyon** (problem türüne göre)
   * Karar ağacı tabanlı basit bir model (Decision Tree)
2. **Orta Seviye Modeller**
   * **Random Forest**
   * **Gradient Boosting** (XGBoost, LightGBM, CatBoost gibi kütüphaneler)
3. **Destek Vektör Makineleri (SVM)**
   * Linear veya RBF çekirdekli SVM ile sınıflandırma veya regresyon.
4. **Yapay Sinir Ağları (Neural Networks)**
   * Basit bir MLP (Multi-layer Perceptron) denemesi (Keras/TensorFlow veya PyTorch).
   * (İsteğe bağlı) Veriniz metinse, LSTM / Transformers gibi modeller.
   * (İsteğe bağlı) Görsel veri varsa, CNN tabanlı basit bir mimari.
5. **Kümeleme (Clustering)** denemeleri (eğer problem yapısı uygunsa)
   * K-Means, DBSCAN veya Hierarchical Clustering gibi yöntemler.

**Not**: Yukarıdaki algoritmalardan birkaçı problemle çok alakalı olmayabilir. Yine de öğrenme amacıyla “farklı tip” modeller denemek ve sonuçlarını karşılaştırmak, bize model çeşitliliği ve avantaj/dezavantaj bakış açısı kazandırır.

**Model Eğitim Süreci**

* **Train-Test (veya Train-Validation-Test) Ayrımı**
  + Veriyi eğitim ve test şeklinde ayırmak (%70-80 train, %20-30 test gibi).
  + Çok veri varsa ek bir **validation** seti veya **k-fold cross validation** kullanmak.
* **Hiperparametre Optimizasyonu**
  + GridSearchCV, RandomizedSearchCV veya Hyperopt gibi yöntemler.
  + Özellikle Random Forest, XGBoost gibi modellerde hiperparametre ayarları önemlidir.
  + Neural network için katman sayısı, nöron sayısı, öğrenme oranı vb.

**Görev**: Her modelin eğitim süreci, kullanılan hiperparametreler ve elde edilen **ara** sonuçlar not edilerek, modelin performans farklılıkları analiz edilsin.

**5) Model Değerlendirme ve Karşılaştırma (Model Evaluation)**

1. **Performans Metrikleri**
   * **Sınıflandırma** için: Accuracy, Precision, Recall, F1-score, ROC AUC, confusion matrix.
   * **Regresyon** için: MSE, RMSE, MAE, R^2.
   * **Kümeleme** için: Silhouette skoru, Davies-Bouldin vb.
2. **Karşılaştırma Tablosu**
   * Tüm modellerin performans metriklerini bir tabloda veya grafikte (bar chart) gösterme.
   * Hangi modelin en iyi sonuç verdiğini ve hangi metrikte daha başarılı olduğunu raporla.
3. **Model Interpretability (Açıklanabilirlik)**
   * Feature Importances (Özellik önem dereceleri)
   * SHAP, LIME veya benzeri kütüphanelerle model kararlarının açıklanması.
4. **Overfitting / Underfitting Tespiti**
   * Eğitim ve test skorları arasındaki farkın izlenmesi.
   * Öğrenme eğrileri (learning curves) incelenebilir.

**Görev**: Değerlendirme metriklerini, her modelin artı ve eksilerini açıkça sunan bir rapor hazırlanması.

**6) Sonuçlar, Raporlama ve Sunum**

1. **Sonuç Raporu**
   * Projenin amacını, veri setini, modeli, elde edilen sonuçları, en iyi ve en kötü performansa sahip modelleri ve olası iyileştirme önerilerini içeren kapsamlı bir rapor (PDF veya web tabanlı).
2. **Görsel Sunum**
   * Model performans grafiklerini (accuracy, error, ROC, vb.) ve önemli özellikleri gösteren grafikleri içeren bir **dashboard** veya **Jupyter Notebook** sunumu.
   * (İsteğe bağlı) Plotly/Dash, Streamlit veya Power BI/Tableau ile interaktif görselleştirmeler.
3. **Dökümantasyon ve Kod Kalitesi**
   * Proje kodlarının GitHub üzerinde **README** dosyası ve açıklayıcı notlarla paylaşılması.
   * Versiyon kontrolü (Git) kullanımı.
   * **Yeniden üretilebilir** (reproducible) bir proje yapısı: “env” dosyası, “requirements.txt” veya conda environment.
4. **Gelecek Adımlar**
   * Daha fazla veri eklemek veya veri kalitesini artırmak.
   * Modeli **real-time** çalıştırma veya API üzerinden entegrasyon yapma.
   * Modeli **ML Ops** yaklaşımıyla versiyonlama, izleme ve güncelleme.

**7) Zaman Çizelgesi ve Teslimatlar**

Aşağıdaki gibi bir zaman planı verilebilir (örnek):

1. **Gün 1 - 3**: Veri seti araştırması, problem tanımı, EDA hazırlıkları.
2. **Gün 4 - 7**: Veri temizleme, özelllik mühendisliği, ilk basit model denemeleri.
3. **Gün 8 - 11**: Farklı model denemeleri (en az 5-6), hiperparametre tuning, karşılaştırma.
4. **Gün 12 - 15**: Nihai model seçimi, sonuçların görselleştirilmesi, rapor dokümantasyonu.
5. **Gün 16 - …**: Sunum hazırlığı, projeyi paylaşma ve geri bildirim toplama.

Bu süre esnetilebilir. Hızlı ilerliyorsanız yeni görevler eklenir yavaş bir ilerleme oluyorsa aralıklar uzatılabilir.

**8) Beklenen Çıktılar (Deliverables)**

1. **Proje Raporu**:
   * Problem tanımı, veri seti ve veri ön işleme adımları, modellemeler ve sonuç karşılaştırmaları, nihai değerlendirme ve geleceğe yönelik öneriler.
2. **Kod Dosyaları**:
   * Jupyter Notebook (veya ilgili IDE’de) açıklamalı kodlar.
   * “requirements.txt” veya “environment.yml” gibi bağımlılık listesi.
3. **Sunum/Demo**:
   * Model performansını ve önemli bulguları özetleyen 10-15 slaytlık bir sunum veya interaktif bir uygulama (Streamlit, Dash, vs.).
4. **GitHub Reposu (Opsiyonel)**:
   * Projenin herkese açık veya şirket içi bir versiyon kontrol sisteminde versiyonlanmış ve açıklamalı hali.

**9) Ek Olası Genişletmeler (İsteğe Bağlı)**

* **AutoML** denemeleri (PyCaret, Auto-sklearn, H2O.ai vb.)
* **Model Pipeline**’larını üretime hazır hâle getiren **CI/CD entegrasyonları** (GitHub Actions, Jenkins vb.)
* **Model izleme (monitoring)** ve **drift tespiti**
* **Transfer Learning** (eğer resim veya metin tabanlı veri varsa)
* **Deep Learning**’in daha ileri seviye uygulamaları (ör. CNN mimarileri, Transformers).
* **A/B Testi** ile modelin sahada gerçek kullanıcı etkileşimleri üzerinde test edilmesi.

**Özet / Kapanış**

Bu proje planı, sizin **kendi seçtiğiniz problem ve veri seti** üzerinde tam bir makine öğrenimi hayat döngüsü deneyimi yaşamanızı sağlayacak şekilde tasarlanmıştır:

1. **Problem Tanımı ve Veri Toplama**
2. **Veri Keşfi ve Görselleştirme (EDA)**
3. **Veri Temizleme ve Özellik Mühendisliği**
4. **Farklı Modeller ile Denemeler (en az 5-6 model)**
5. **Değerlendirme ve Karşılaştırma**
6. **Raporlama, Sunum ve Paylaşım**

Her aşama sonunda çıktıları beraber inceleyeceğiz. Projenizin hızlı bitmesi durumunda seçmediğiniz diğer iki problem türü üzerine de aynı adımlar izlenerek birer proje yapılacaktır. Örneğin, bir sınıflandırma problemi üzerine proje yaptıysanız ve bu proje beklenenden erken bittiyse regresyon ve kümelendirme problem türleri üzerine aynı adımları takip ederek projeler geliştirilecek.