Proje Ana Alanı : **Yazılım**

Proje Tematik Alanı : **Halk Sağlığı ve Koruyucu Sağlık Hizmetleri**

Proje Adı (Başlığı) : **Kan Sayımı Sonuçlarından Yapay Zeka İle Rahim Kanseri Tespiti.**

Özet:

Son zamanlarda popülerleşen yapay zeka trendini Türkiye'nin de yakalamasını, rahim kanserinde tanı ile erken tanıda basit ve efektif bir yöntem ile kadınlarımızın hayat konforunun arttırılması hedeflenmiştir. Proje bu hedeften yola çıkarak, son halinde %95.8 isabet oranına kadar erişmiştir. Projemizin ara yüzü ile birlikte hastanelerimize çok kolay entegre edilebilir hale getirilmiş, kullanımı ve ulaşımı basitleştirilmiştir. Projemiz Tasarım ve Geliştirme Araştırma yöntemi kullanılarak oluşturulmuş olup, bir tanı yöntemi olarak kullanılmasını hedefliyoruz. Rahim ağzı kanserinin aksine, rahim kanseri, tanısı çok daha zor olan ve tanı yöntemi az olan bir hastalıktır. Ölüm riski yüksek olan bir kanser türüdür, 4.aşamadaki bir rahim kanserinin ölüm oranı %85 olmakla birlikte, bu projemiz ile kanser bu kadar ilerlemeden basit bir kan testi ile bunun keşfi, ve tedavisini mümkün kılmayı hedefliyoruz. Projemizde beyaz kan hücrelerinin kanserden etkileneceği düşüncesinden yola çıkarak yazdığımız algoritma ile, aslında bağlantısız olduğu düşünülen başka kan değerlerinin de bu kanserden önemli ölçüde etkilendiği tespit edilmiştir. Hastaların hayat konforunu arttırmak, kanser ilerlemeden teşhis koymak ve buna karşı önlem almak için geliştirdiğimiz yapay zeka ile hedefimize vardığımızı söyleyebiliriz.

Anahtar Kelimeler:

Rahim kanseri, Yapay Zeka, Kan Sayımı, Makine Öğrenmesi, Python

Amaç:

Sorunumuz olarak belirlediğimiz Rahim kanserinde, şu anki tespit metodlarından farklı olarak biz ne yapabiliriz diye düşündük. Amacımız, her kadınımızın farkında olmasa bile yakalanabileceği, beklide toplum baskısı yüzünden dile getiremeyen kadınlarımızın yaşam kalitesini arttırmak. Yapay zeka ile basit bir kan sayımı sayesinde, şu anki teşhis yöntemleri dışında, onlardan farklı olarak kullanımının basit oluşu, her kan sayımında testinin yapılabilmesi, MRI ve Biyopsi gibi tespit yönlerinin yanında herkese ulaşılabilir olması bizim avantajımızdır. Kanser ilerlemeden erken evrede teşhis yapabilmekti bizim amacımız. Makineye öğretilen kan verileri ile kanserden etkilenen değişkenleri, var sayılan bağımlı değişkenlerin dışında, makinelerin ve yapay zekanın gücünü kullanarak doktorlarımızın gözünden kaçan, insanların aksine matematiksel olarak tahmin yaptırıp, bize sonuç verebilen bir yapay zeka geliştirmekti amacımız, ve başardık. Tıp literatüründe değişiklilik yaratabilecek kandaki farklı değerlerinde bunlardan etkilendiğini keşfetmekti hedefimiz yazılım bilgilerimiz ve yapay zekayı kullanarak. Hedefimiz olarak basitleştirilmiş bir test yöntemi bulmak, ve her yere ulaşmaktı. Bunun yöntemini ise son zamanlarda popülerleşen yapay zekada bulduk.

Giriş:

Günümüzde hayatımızın iyice içine giren yapay zeka, son zamanlarda kendini her alanda göstermeye başladı. Örnek olarak sağlık alanında görüntü işleme ve yapay zeka ile teşhis konulan meme kanseri gibi, yada en basitinden konuşma modelleri, ve hatta her telefonda bulunan asistanlar gibi. Yapay zeka bu kadar popüler olmuşken biz de bundan yaralanmak istedik. Yapay zeka ile sağlık arasındaki yetersizliği kendimize amaç edinip onun üzerinden yürümeye karar verdik. Bunu nasıl pratik ve gündelik hayata entegre edebiliriz sorusu ise bize beyin fırtınası yaptırmaya yöneltti. Endüstriyel ölçekte ve kullanılabilecek işlevsel bir yapay zeka veya makine öğrenmesi yazmanın zorluğunu göz önüne alarak, hastaların hayat konforunu yapay zeka ile nasıl arttırabilirize çevirdik sorumuzu. İşte bu noktadan yola çıkarak, kadınlarımızın hayatını riske atan, tehlikeli bir kanser türü olan rahim kanseri ve teşhis yöntemlerindeki eksiklikler dikkatimizi çekti.

Biz bu projemizde hipotezimizi kanser hücreleri ile savaşan beyaz kan hücreleri sayısının, kansere yakalanan hastalarda artması gerektiğini düşünerekten nasıl bunu tespit edebiliriz ve şu anda kullanılan teşhis yöntemlerinden nasıl daha basit bir yöntem yaratabiliriz olarak belirledik. Uygulamamızda Python yazılım dilini bilimsel olarak kullanılabilmesini sağlayan Pandas ve Numpy modülünü kullandık. Bu modüllere ek olarak Kocaeli Üniversitesinden temin ettiğimiz verileri test ve makineye öğretme olarak ayırmamıza yardımcı olan Sklearn modulünden bir kısmını da Pythonun kendisiyle birlikte gelen modülleri ile birlikte kullandık. Bu modüllerden ve matematik bilgilerinden faydalanarak, bir KNN(K-Nearest Neighbor) algoritması oluşturduk. KNN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır. Lazy kavramını anlamaya çalışırsak, eager learning aksine lazy learning'in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini "ezberler". Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar. Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır. Bu sayede makineye test için gönderilen verilerin bir tahmin yapmasını ve bize bir sonuç göndermesi sağlanır.

Rahim kanseri ile yapay zeka ilgili yapılan araştırmaların az olması bizi yönlendirdi. Daha önce yapılan araştırmalarda sadece basit risk faktörleri alınaraktan yapıldığı için sadece kansere yakalanma "risk" tespiti yapabiliyordu. Bizim başardığımız ise basit bir kan sayımı ile kanser olup olmadığını tahmin etmek. Yapılan bir diğer araştırmada ise görüntü işleme, MRI sonuçları gibi teknikler kullanıldığı için bizim ortaya çıkarttığımız ürünün uygulanabilirlik ve basitliği ile öne çıktığını söyleyebiliriz.

Bahsi geçen araştırmaların en büyük ortak problemi, uygulanmasının zor oluşu ve düşük bir isabet oranı geri getirmesi. Bizim projemizde %95 e kadar çıkan isabet oranımızla birlikte tembel öğrenme (lazy learning) yapan bir model seçmemiz bunu çalışması için güçlü makineler gerektirmeyen, basit ve gerçekçi bir uygulama yöntemi önümüze sunuyor. Doktarlara destek olacak bu sistemin gerçekçi bir perspektiften bakılarak bunun erişilebilirliği, birçok kadınımızın hayatını kurtaracak, ulaşımı kolay ve makinelerin üstüne düşük stres bindiren bu uygulama ile Türkiye'nin her tarafına yayılıp, toplum baskısı altında

kalmış ve bunu dillendiremeyen vatandaşlarımızın da hayat konforunu arttırmak ve bu hastalıktan kurtarmak bizi daha önceki araştırmalardan ön plana atmaktadır.

Yöntem:

Projemizde Tasarım ve Geliştirme Araştırma yöntemlerini kullandık. Yapay zeka ile rahim kanseri tespiti projemizi geliştirirken teme olarak Python yazılım dili ve Python'un bilimsel olarak kullanılmasını sağlayan NumPy, Pandas ayrıca verilerimizi test ve öğretim olarak ikiye ayıran Sklearn modülünden "train_test_split()" fonksiyonundan yaralandık. Bunun yanında Pythonla birlikte gelen ana kütüphanelerden os, warnings, collections'u kullandık. Makine öğrenmesi algoritması olarak ise türlü sebeplerden kolayı KNN(K-Nearest Neighbor) algoritmasını tercih ettik. Database için hafif olması sebebiyle sqlite, web arayüzü için ise Flask modülünden faydalandık.

Projeyi önce Kocaeli Üniversitesinden etik onayı ile aldığımız verileri düzenleyerek başlıyoruz. Bu düzenleme için pandas kütüphanesini kullanıyoruz. Pandasın bizim verilerimiz için oluşturduğu Dataframe objesi üzerinde kullandığımız "replace()" metodu ile verilerdeki belirtilmeyen, boş olarak girilen "-" ve "----" değerlerini 0 ile değiştiriyoruz. Sonrasında "string" veri tipinde olan verileri "float" veri tipine yazdığımız "ConvertFloat()" fonksiyonunu Pandas yardımı ile verilerimize "apply()" fonksiyonu ile uyguluyoruz. Sonrasında ara gurup olarak bilinen "EIN" grubunu algoritmanın kafasını karıştırdığı için veri setimizden çıkarıyoruz (Ein veri grubu yaklaşık %5-%3 sapmaya sebep olmaktadır). Ardından hastaların kanser durumuna göre onların verilerine bir numerik değer atayarak makinenin bu değerler üzerinden tahmin etmesini sağlamak için "map()" fonksiyonu kullanılaraktan bu değerler atanıyor. Veriyi düzenlemenin son aşaması olarak makineye öğretilecek değerler "X" ve sonuç olarak ulaşması gereken değerler "y" olarak tanımlanıyor.

```
81  path = os.getcwd()+r"\Datasets\default.csv"
82  df = pd.read_csv(path)
83  df.replace("-", "0.0", inplace=True)
84  df.replace("----", "0.0", inplace=True)
85
86  def ConvertFloat(value):
87      return float(value.replace(',', '.'))
88
89  for column in df.columns[1:]:
90      df[column] = df[column].apply(ConvertFloat)
91
92  df = df[df.status != "ein"]
93
94  status = {"basitler": 0,"ein":2, "highrisk": 1, "lowrisk": 1}
95  yUnmapped = df['status']
96  y = yUnmapped.map(status)
97  X = df.drop(columns=['status',"NRBC"])
```

Sonrasında ise kodumuzun isabet oranını almak için onu birden çok çalıştıran bir "for" döngüsü içine alıyoruz. Bu döngü içinde önceden yazdığımız KNN algoritmasını tanımlayıp, verilerimizi "makine isabeti" ve "test isabeti" olarak ayırıyoruz. Sonrasında parametrelerin önemlerine göre onların bazılarını kombinasyonlar oluşturacak şekilde tanımlıyoruz. Bazı verilerin tekil isabetlerini de almak için öncelikle içinde "_" bulunan kan değerlerini oran olduğu için çıkarıyoruz. Sonrasında belirlediğimiz parametreleri "TestParameters" listesine ekleyip veri ayırmayı ve parametre

düzenlemeyi burada sonlandırıyoruz. Verilerin ayrılmasında Sklearn'ın "train_test_split()" modülünden faydalanıyoruz. Ayrıca bu aşamada basit birkaç Python'un fonskiyonunu da kullanıyoruz.

```python
for rs in range(1000):
    KnnAlgorithm = Knn(config["n_neighbor"])

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = config["test_size"],)

    X_hand, X_machine, y_hand, y_machine = X_test.iloc[:len(X_test)//2], X_test.iloc[len(X_test)//2:], y_test.iloc[:len(y_test)//2], y_test.iloc[len(y_test)//2:]

    TestParameters = [
        {
            "params":["MCHC", "RDWSD", "RDWCV", "PCT", "PDW"],
            "name":"Comb-1"
        },
        {
            "params":['WBC', 'MCV', 'MCHC', 'RDWSD', 'RDWCV', 'PCT', 'PDW'],
            "name":"Comb-2"
        },
        {
            "params":['EOS', 'BASO', 'WBC', 'MONO', 'HCT', 'MCHC', 'RDWSD', 'RDWCV', 'MPV', 'PCT', 'PDW'],
            "name":"Comb-3"
        },
        # {
            # "params":['LYM', 'BASO', 'HGB', 'HCT', 'MCV', 'MCHC', 'RDWSD', 'RDWCV', 'PCT', 'PDW'],
            # "name":"Comb-4"
        # }
    ]

    all_columns = X.columns.tolist()
    columns = all_columns[:]
    columnsForAppending = ['LYM', 'BASO', 'HGB', 'HCT', 'MCV', 'MCHC', 'RDWSD', 'RDWCV', 'PCT', 'PDW']

    for col in columns:
        if "_" in col:
            columns.pop(columns.index(col))

    for cosCol in columnsForAppending:
        TestParameters.append({"params":[cosCol], "name":cosCol})
```

Ardından asıl makine öğrenmesi kısmı geliyor. Burada kendi yaptığımız özelleştirilmiş oylama sistemi kullanıyoruz. Bu oylama sisteminde her kombinasyon önce "KNN" sınıfının bir fonksiyonu olan "TrainKnn()" ile her kombinasyonun isabet oranı alınıp "TestParameters" listesinde kendi kombinasyonunun yanına kaydedeliyor. Ardından 2. aşama teste geçiliyor. Burada yine "KNN" sınıfının bir fonksiyonu olan "TestKnn()" ile her bir test verisinin her parametrelerin önerdiği cevap ile o parametre kombinasyonlarının isabet oranı "ResultsOfTests" listesine kaydedilip, sonrasında bu değerler alınaraktan eğer sonucun kanser olduğunu düşünüyorsa, isabet oranının üssü ikisi alınarak oyladığı değere ekleniyor. Sonrasında doğru tahmin ettiyse eğer "gotRight" 1 arttırılıyor, eğer yanlış ise "gotWrong" 1 arttırılıyor.

```python
    for parameters in TestParameters:
        acc = KnnAlgorithm.TrainKnn(parameters["params"])
        parameters["acc"] = acc*config["accMultiplier"]

    gotRight = 0
    gotWrong = 0

    for i,useless in enumerate(X_hand):
        x_var = X_hand.reset_index(drop=True).iloc[i]
        y_var = y_hand.reset_index(drop=True).iloc[i]

        resultsOfTests = []
        for params in TestParameters:
            resultsOfTest = {}
            pred = KnnAlgorithm.TestKnn(params["params"],x_var,y_var)
            resultsOfTest["accModel"] = params["acc"]
            resultsOfTest["pred"] = pred[0]

            resultsOfTests.append(resultsOfTest)

        voteValue0 = 0
        voteValue1 = 0
        voted = 0

        for resTest in resultsOfTests:
            if resTest["accModel"] >= config["modelAffectionLimitator"]:
                if resTest["pred"] == 1:
                    voteValue1 = voteValue1 + resTest["accModel"]**config["accResultMultiplier"]
                else:
                    voteValue0 = voteValue0 + resTest["accModel"]**config["accResultMultiplier"]

        if voteValue1 > voteValue0:
            voted = 1
        else:
            voted = 0

        if voted == y_var:
            gotRight += 1
        else:
            gotWrong += 1
```

En son ise toplam veri ile doğru bilinen veriler birbirine bölünerek isabet oranı hesaplanıyor. Bu değer "overallscores" listesine kaydediliyor, ve "for" döngüsünden

çıkıldıktan sonra bütün bu testlerin getirdiği isabet oranı toplam isabet oranına bölünüp ortalama isabet değeri ve en iyi isabet değeri bulunuyor.

```
181         score = gotRight/(gotRight+gotWrong)
182         overallscores.append(score)
183         print(f"% {round(score,5)*100}")
184         if score > BestScore and BestScore != 1 :
185             BestScore = score
186 
187     scTotal = 0
188     for sc in overallscores:
189         scTotal+=sc
190 
191     totalScores = scTotal = scTotal/len(overallscores)
192 
193     print("Best Accuracy: %",round(BestScore,5)*100)
194     print("Average Accuracy: %",round(totalScores,5)*100)
```

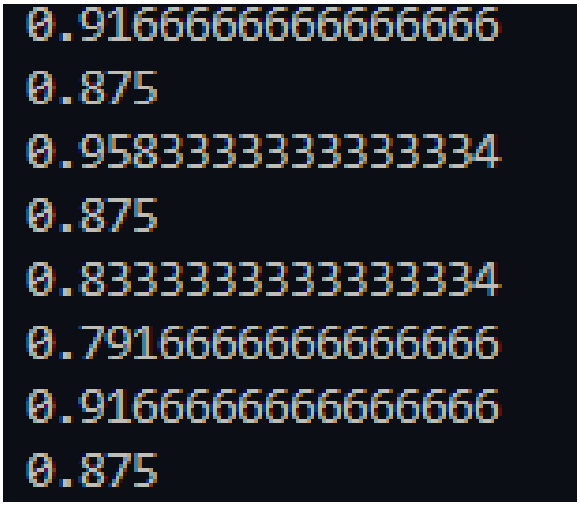
Algoritmamızın dışında tanımladığımız “KNN” sınıfında ise “fit()” fonksiyonu modelin içine verileri yerleştirmemize, “TrainKnn()” fonksiyonu isabet oranı almaya, “TestKnn()” fonksiyonu gelen veriler için tahminde bulunmaya, “CalculateAccuracy()” fonksiyonu isabet hesaplama görevlerini üslenirken, “predict()” ve “getDistace()” fonksiyonu asıl tahmin ve makine öğrenmesini gerçekleştiren fonksiyonlardır.

```
class Knn:
    def __init__(self, k):
        self.k = k
        self.X = None
        self.y = None

    def getDistance(self, p, q):
        p = np.array(p)
        q = np.array(q)

        if p.shape != q.shape:
            raise ValueError("Input arrays p and q must have the same shape.")

        return np.sqrt(np.sum((p - q) ** 2))

    def fit(self, X, y):
        self.X = X.values
        self.y = y.values

    def predict(self, X_test):
        y_pred = []

        for new_point in X_test:
            distances = []
            for i, point in enumerate(self.X):
                distance = self.getDistance(point, new_point)
                distances.append([distance, self.y[i]])

            categories = [category[1] for category in sorted(distances)[:self.k]]
            result = Counter(categories).most_common(1)[0][0]
            y_pred.append(result)

        return y_pred
```

```
50      def CalculateAccuracy(self, machine, prediction):
51          machineData = machine.values
52          gotRightAcc = 0
53
54          for index in range(len(machineData)):
55              if machineData[index] == prediction[index][0]:
56                  gotRightAcc += 1
57
58          return gotRightAcc/len(machineData)
59
60      def TrainKnn(self,selected_features):
61          selected_columns = list(selected_features)
62          self.fit(X_train[selected_columns], y_train)
63          y_pred = []
64          for i in range(len(X_machine)):
65              x_var = X_machine[selected_columns].iloc[i]
66              y_pred.append(self.TestKnn(selected_features, x_var, y_machine.iloc[i]))
67          accuracy = self.CalculateAccuracy(y_machine,y_pred)
68
69          return accuracy
70
71      def TestKnn(self, selected_features, x_var, y_var):
72          selected_columns = list(selected_features)
73          self.fit(X_train[selected_columns], y_train)
74          x_var_2d = x_var[selected_columns].values.reshape(1, -1)
75          y_pred = self.predict(x_var_2d)
76
77          return y_pred
```

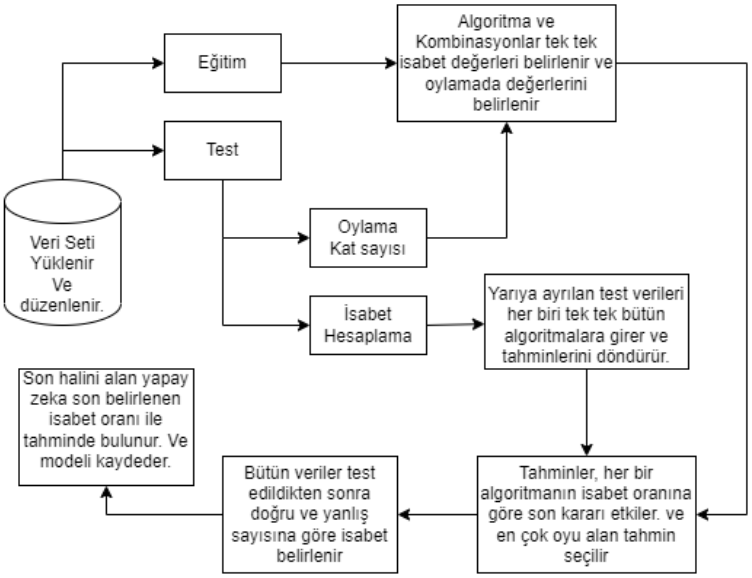
KNN(K-Nearest Neighbor) algoritması oluşturduk. KNN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır. Lazy kavramını anlamaya çalışırsak, eager learning aksine lazy learning'in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini "ezberler". Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar. Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır. Bu sayede makineye test için gönderilen verilerin bir tahmin yapmasını ve bize bir sonuç göndermesi sağlanır. Kodun zamansal karmaşıklığı ise O(N*M), N test verisinin büyüklüğünü, M ise öğretim verisinin büyüklüğünü ifade ediyor. Yanında ise ekstradan döndüler bulunmaktadır.

En sonunda hesaplanan isabet oranı ise şu şekilde gözükmektedir (1 değeri %100 isabet, 0.9 ise %90 isabet demektir) :



Web arayüzü, örnek kan sonucu ve algoritmanın basitleştirilmiş hali ise şu şekildedir:

resim-1



resim-2

Test Adı	Sonuç	Durum	Birim	Referans Değerler	Önceki Sonuçlar
Hemogram					
WBC (Lökosit)	5,16		x10^3/µL	3,46 - 10,04	7,33 (05/05/22) 7,97 (01/04/22)
NEU (Nötrofil Sayısı)	2,910		x10^3/µL	1,47 - 7,34	4,610 (05/05/22) 4,780 (01/04/22)
NEU % (Nötrofil Yüzdesi)	56,4		%	42,7 - 73,2	62,9 (05/05/22) 60,0 (01/04/22)
LYM (Lenfosit Sayısı)	1,700		x10^3/µL	1,05 - 3,17	2,070 (05/05/22) 2,510 (01/04/22)
LYM % (Lenfosit Yüzdesi)	32,9		%	21,6 - 48,5	28,2 (05/05/22) 31,5 (01/04/22)
MONO (Monosit Sayısı)	0,380		x10^3/µL	0,25 - 0,95	0,490 (05/05/22) 0,530 (01/04/22)
MONO % (Monosit Yüzdesi)	7,4		%	4,2 - 13,5	6,7 (05/05/22) 6,6 (01/04/22)
EOS (Eozinofil Sayısı)	0,150		x10^3/µL	0,03 - 0,29	0,140 (05/05/22) 0,130 (01/04/22)
EOS % (Eozinofil Yüzdesi)	2,9		%	0,6 - 5,2	1,9 (05/05/22) 1,6 (01/04/22)
BASO (Basofil Sayısı)	0,020		x10^3/µL	0,02 - 0,07	0,020 (05/05/22) 0,020 (01/04/22)
BASO % (Basofil Yüzdesi)	0,4		%	0,2 - 1,4	0,3 (05/05/22) 0,3 (01/04/22)
RBC (Eritrosit)	3,38	D	x10^6/µL	3,87 - 5,62	3,91 (05/05/22) 4,08 (01/04/22)
HGB (Hemoglobin)	9,80	D	g/dL	12,1 - 16,6	11,50 (05/05/22) 12,30 (01/04/22)
HCT (Hematokrit)	29,5	D	%	36,9 - 52,9	33,7 (05/05/22) 35,0 (01/04/22)
MCV (Ortalama Eritrosit Hacmi)	87,30		fL	81,8 - 98	86,20 (05/05/22) 85,80 (01/04/22)
MCH (Ortalama Hücre Hemoglobin)	29,00		pg	25,6 - 32,3	29,40 (05/05/22) 30,10 (01/04/22)
MCHC (Ortalama Hücre Hemog.Konsant.)	33,20	Y	g/dL	28,2 - 31,7	34,10 (05/05/22) 35,10 (01/04/22)
RDW-SD	43,10		fL	38 - 50	44,80 (05/05/22) 48,20 (01/04/22)
RDW-CV	13,60		%	11,2 - 14	14,40 (05/05/22) 15,40 (01/04/22)
PLT (Trombosit)	256		x10^3/µL	172 - 380	296 (05/05/22) 297 (01/04/22)
MPV (Ortalama Trombosit Hacmi)	10,90		fL	9,2 - 12,2	10,10 (05/05/22) 10,50 (01/04/22)
PCT (Platekrit)	0,28		%	0,19 - 0,41	0,30 (05/05/22) 0,31 (01/04/22)
PDW (Trombosit Dağılım Genişliği)	12,20		fL	9,5 - 15,5	10,90 (05/05/22) 11,40 (01/04/22)
NRBC	0,00		x10^3/µL	0 - 0,015	0,00 (05/05/22) 0,00 (01/04/22)
NRBC %	0,0		%	0 - 0,029	0,0 (05/05/22) 0,0 (01/04/22)

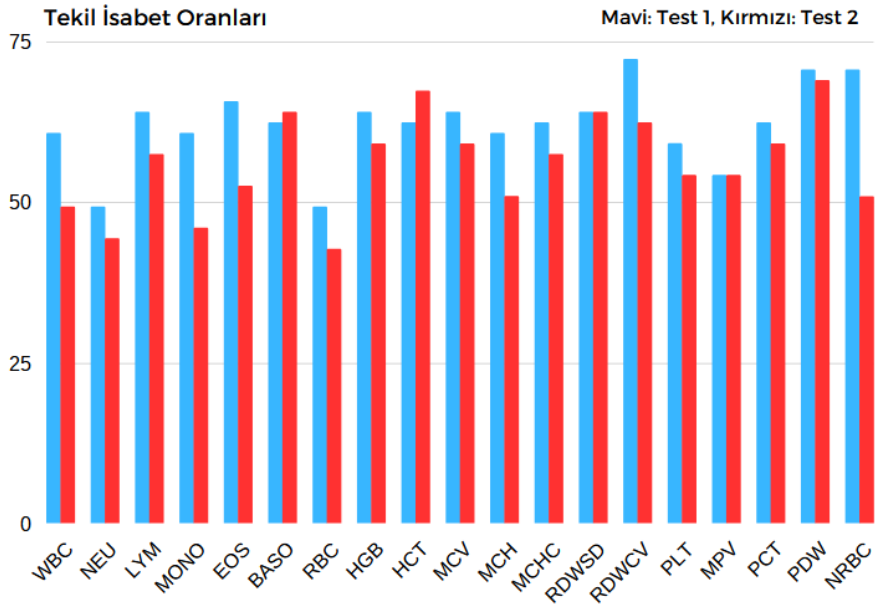
resim-3

Proje İş-zaman Çizelgesi:

Aylar								
İşin Tanımı	Nisan	Mayıs	Haziran	Temmuz	Ağustos	Eylül	Ekim	Kasım
Literatür Taraması	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Verilerin Toplanması	**X**	**X**	**X**					
Yapay Zeka, Algoritma Ve Arayüzün Geliştirilmesi			**X**	**X**	**X**	**X**		
Proje Raporu Yazımı						**X**	**X**	**X**

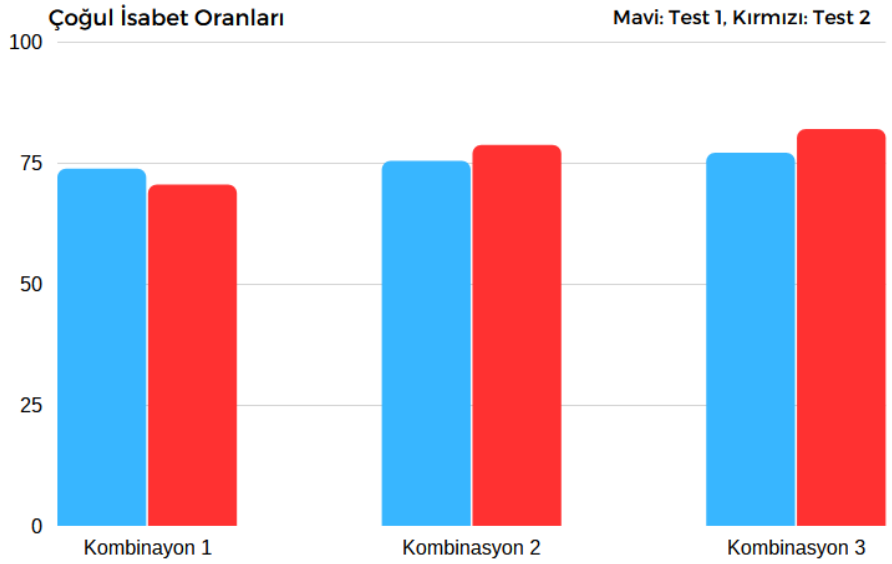
Bulgular:

Projemizin isabet oranı ve kullanılabilirliğini gözlemlemek adına Kocaeli Üniversitesinden temin ettiğimiz verilerde 603 hastanın verisi üzerinde çalışıp, bunun yaklaşık 60 tanesini test amaçlı kullanılmıştır. Kan değerlerinin tekil isabet oranı(grafik-1) ve kombinasyonlarının isabet oranı(grafik-2) ise şu şekildedir.



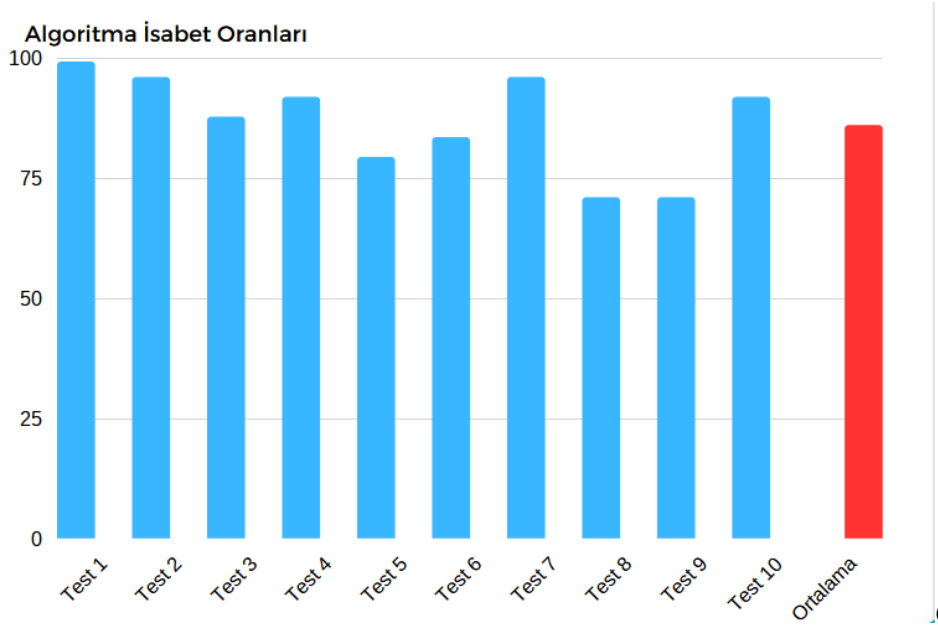
Grafik-1

Kombinasyon 1: "MCHC", "RDWSD", "RDWCV", "PCT", "PDW"
Kombinasyon 2: "WBC", "MCV", "MCHC", "RDWSD", "RDWCV", "PCT", "PDW"
Kombinasyon 3: "EOS", "BASO", "WBC", "MONO", "HCT", "MCHC", "RDWSD", "RDWCV", "MPV", "PCT", "PDW"
Tekil Kullanılan Değerler: "LYM", "BASO", "HGB", "HCT", "MCV", "MCHC", "RDWSD", "RDWCV", "PCT", "PDW"



Grafik-2

Bunlar algoritmamız içinde kullandığımız ve tekil olarak kullandığımız değerler olmak üzere tahminimizin aksine beyaz kan hücrelerinin değerleri bağımsız sanılan değerlerden daha isabetli çıkmıştır. Bunları kullanarak kombinasyonlar ve değişken önemi sıralanmıştır (Feature Selection). Algoritmamızın bunların hepsini göz önüne alarak yaptığı tahminlerin isabet oranı ise şu şekildedir(grafik-3):



Grafik-3

Grafik-3 de de görüldüğü gibi algoritmamız düşük değerli isabet oranlarını geride bırakarak farklı kombinasyonlar ve üst üste konulmuş KNN tahminleri ile bu sonuca ulaşmıştır (Grafik-3 ortalama değer 1000 test sonucu elde edilen değerdir.)

Sonuç ve Tartışma:

Elde ettiğimiz bulgulara göre , rahim kanserinin kan sonucunu etkilediği, yapay zekanın sağlıkta kanser alanında kullanılabileceği ve bağımsız olduğu düşünülen kan değerlerinin aslında bağımlı olduğunu, önemli bir etkide bulunduğu tespit edilmiştir. Algoritmamız %95 isabet oranına kadar doğru tahmin yapabilmekte olup, basit bir kan sonucu ile doktorlarımıza destek çıkacak bir sistem üretilmiştir. Aynı zamanda kurulumunun basitliği, ihtiyaç duyduğu makine gücünün düşüklüğü ile birlikte yüksek uygulanabilirliğe sahip oluşu ülkemizin 4 bir köşesine yayılıp azınlıkta kalan ve bunu dillendiremeyen kadınlarımızın teşhisinde ve aynı zamanda doktorlara destek olabilecek bir ürün ortaya konmuştur. Masrafının olmaması bunu her kan sayım sonucunda uygulanabilecek bir test haline getiriyor.

Projemizdeki kodun esnek olması sebebiyle belli bir grup kansere yönelik veriler girilerek o kansere sahip olma risk oranı tespit edilebilmesi ile birlikte eğer veri seti hastaneler veya sağlık bakanlığı tarafından genişletilirse daha isabetli tahminler yapılabilecektir.

Kurulum için sadece kodun çalıştırılıp web arayüzünden değerlerin girilmesi yeterlidir. Sadece hastanelerde 1 tane bilgisayar ile bu kod çalıştırılabilmesiyle birlikte eğer hastanelerin ana sunucusuna dahil edilebilirse bu test otomatikleştirilebilip daha çok tanıda bulunabiliriz.

Özellikle bu projemizin iç kesimlerde kalan ve batıl inançlarına bağlı yaşayan kadınlarımızın baskı ve utanç altında olmadan basit bir şekilde sadece kan ile teşhis konulabilmesi ve bu şekilde farkında olmadan kanserin gelişimini önlemeye çalışmanın bir çok insanı kurtarabileceğini düşünüyoruz. Bu yönden sağladığı basitlik ile sağlık için olsa bile doktorlardan çekilen ve muayeneyi reddeden hastalarımız için yardımcı olabileceğiz

Projemizi daha geniş bir veri yelpazesinde test edemesek de, eğer sadece belli bir kanser türüne ait geniş bir veri seti ile birlikte başka kanser türlerinden şüphelenilen hastalarda doktorun isteği üzerine test yapılması ve doktorların işini basitleştirebileceğini düşünüyoruz.

Projemizin bir diğer çarpıcı bulgusu ise beyaz kan hücrelerinin kanser ile savaştığı düşüncesiyle yola çıktığımızda fazla ilerleyemediğimizi fark etmiştik. İsabet oranın %60 ı geçemiyordu. Bu bize başka bağlı değişkenler olabileceği düşüncesine itti. Kan değerlerinde sadece beyaz kan hücrelerinin değerlerinin aksine başka kan değerlerinin de bundan etkilenmesi, kombinasyonlarda öne çıkmaları ve tekil değerlerinin de yüksek olması bu alanda yapılabilecek başka araştırmalara da yol açıyor. Tıp literatürüne kanser gibi önemli bir konu ile ilgili yeni bir bilgi ekleyebilecek bir projeye öncü olabilecek bir tespit yaptığımızı düşünüyoruz.

Öneriler:

Geliştirdiğimiz yapay zeka ve algoritmada tespit ettiğimiz bağımlı değişkenlerin araştırılması önemli olabilir. Bunu yanında veri setinin genişletilmesi daha isabetli tahminlerle sonuçlanmasının yanında, eğer bilgiler arasına yaş, kilo, sigara ve alkol tüketimi gibi veriler eklenir ise %85 ortalamadan %95 e kadar çıkabileceğine inanmaktayız. Alanında uzman olan kişilerle çalışıp algoritma geliştirilebilir ve her hastanenin lokal olarak web arayüzünü çalıştırmaktansa ortak bir veri toplanma noktası oluşturulup, her hastane bu ortak olarak yerleştirilen web sitesi üzerinden veri ekleyip, hastalar için teşhiste kullanılabilir. Algoritma

farklı kanser verilerine uyarlanabilir ve etki alanı genişletilebilir. Fakat halka açık hale getirilmesinin sonucu insanların kendi kendine teşhis koymasıyla sonuçlanabileceğinden sadece hastanelere erişim vermeyi doğru buluyoruz. Alanında uzman insanlar ile birlikte yapılacak eklentiler ile gelişebilecek bir proje.

Kaynakça:

Ölüm oranı: https://www.cancerresearchuk.org/about-cancer/womb-cancer/survival#:~:text=Survival%20for%20all%20stages%20of%20womb%20cancer&text=90%20out%20of%20every%20100%20(90%25)%20survive%20their%20cancer,for%205%20years%20or%20more
Araştırma 1: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9860482
Araştırma 2: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9365068

Etik Onayı: