Monte Carlo ile Petrol fiyat tahmini

Barış Sanlı , barissanli.com (barissanli.com)

Petrol fiyatlarını sayısal yöntemlerle tahmin etmek mümkün değil ama bazı dönemsel hareketlerin mekaniğini bilmek faydalı olabilir. Aşağıdaki kod dizisinde, öncelikle 2020'nin ilk çeyreğinin neden farklı bir dönem olduğuna bakıyoruz, daha sonra tarihsel dönemleri örnek alırsak petrol fiyatlarının gidebileceği yönleri görmeye çalışıyoruz.

Yapay zeka ile tabii ki 5-6 satırda bir tahmin yapılabilir. Ama biraz daha mantık setini anlamak için böyle bir model faydalı olabilir.

Bölüm 1 - İstatistiksel Bakış

Önce gerekli kütüphanelerimizi yükleyelim.. Alttaki kod ile de grafikleri daha büyük görmek için gerekli olan dizilimi görmekteyiz

In [1]:
```
%pylab inline
import pandas as pd
import seaborn as sb
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
plt.rcParams['figure.figsize'] = [10, 7]  #grafikleri daha büyütmek için
```

Verilerimizi her Çarşamba güncellenen ABD Enerji Bakanlığı spot Brent fiyatlarından alacağız. Bu veriler Bloomberg-Reuters'de gördüğünüz verilerden farklı, çünkü spot fiyatlar. TV ekranlarında gördüğünüz ise genelde gelecek ay, yani teslimi 30-40 gün içinde yapılacak ürünler.

In [3]:
```
link='https://www.eia.gov/dnav/pet/hist_xls/RBRTEd.xls'
```

Şimdi verimizi yükleyelim, "Data 1" çalışma sayfasını seçip, verinin başladığı satırdan veriyi okuyalım

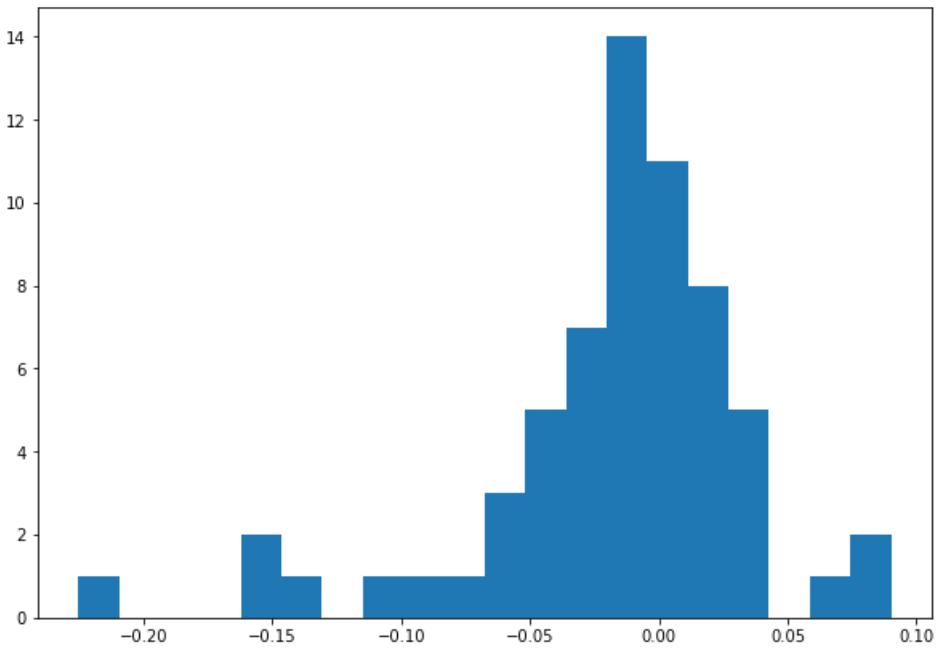
In [4]:
```
data = pd.read_excel(link,'Data 1', skiprows=(0,1))
```

İlk çeyrek verileri 1,2,3.ay verileri olduğundan, 2020 yılının ilk 3 ayının verilerini tüm veri setinden ayıralım

In [5]:
```
data_s=data[(data["Date"].dt.year==2020) & (data["Date"].dt.month<4) ] #2020 i
lk çeyrek
```

Şimdi de bu verimizin histogram-yani istatistiksel dağılımını çıkaralım

In [6]:
```
hist(data_s["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_change().dropna(), bins=20);
```

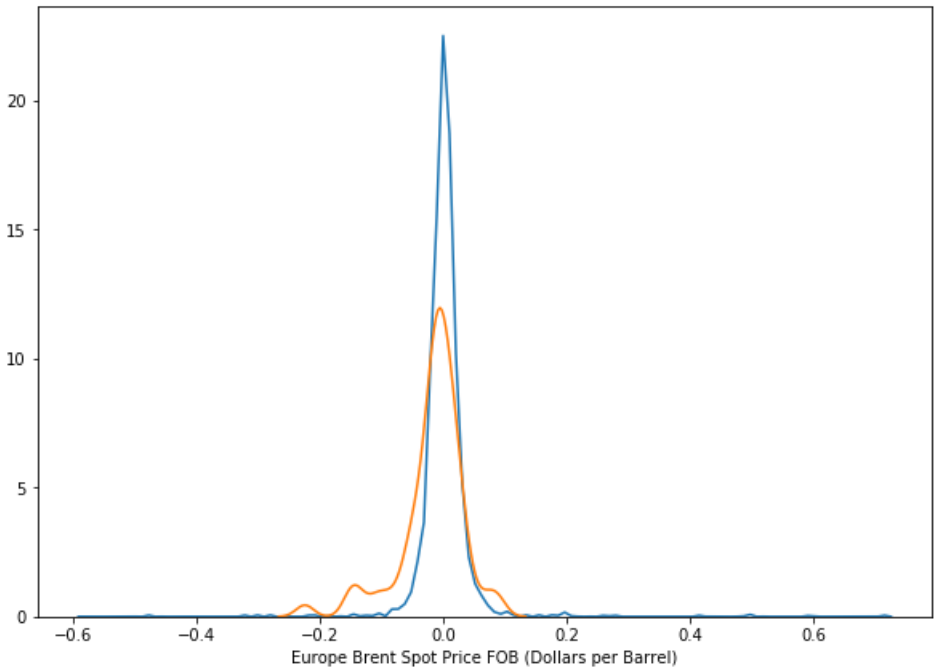


Peki **2020** daha önceki yıllardan ne kadar farklıydı. Şimdi de 2020 öncesi yılların ilk çeyrek verilerini çıkaralım

In [7]:
```
data_yil=data[(data["Date"].dt.year<2020) & (data["Date"].dt.month<4) ]   #2020'den önceki yılların ilk çeyrekleri
```

Şimdi 2020 yılı ilk çeyrek ve 2020 yılı öncesindeki yılların ilk çeyrek değişim oranlarını çizdirelim

In [8]:
```
sb.distplot(data_yil["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_c
hange().dropna(), hist=False, rug=False,hist_kws={"range": [0,1]})
sb.distplot(data_s["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_cha
nge().dropna(), hist=False, rug=False, hist_kws={"range": [0,1]})
#sb.title("2020 yılı ve daha önceki yıllarda ilk çeyrek petrol fiyatı değişim
oranı")
plt.show()
```



Bölüm 2 - Fiyat seviyelerine göre artış oranı mı yüzdesi mi?

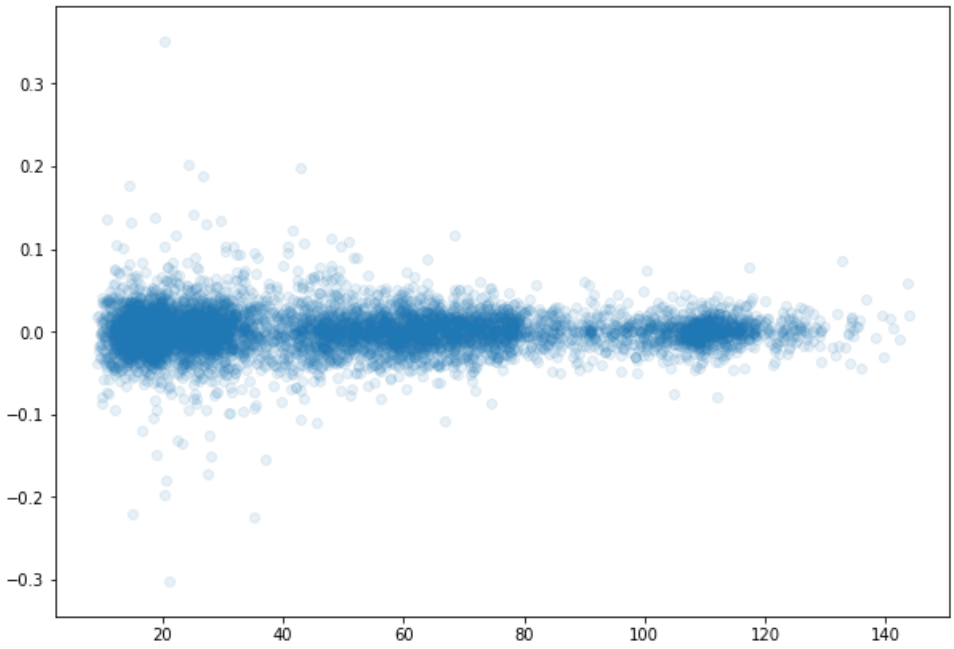
Mesela petrol fiyatları daha düşük iken, petrol artışları % olarak daha büyük olabiliyor şimdi bunu incelemeye çalışalım. Önce yeni bir sütun açarak yüzde değişim oranlarını yerleştirelim

In [9]:
```
data["pct"]=data["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_chang
e()
```

Fiyat seviyelerine göre fiyat değişim % oranlarına, yarı saydam noktalar bütünü olarak bakalım

In [10]:
```
scatter(data["Europe Brent Spot Price FOB (Dollars per Barrel)"],data.pct,alpha=0.1)
```

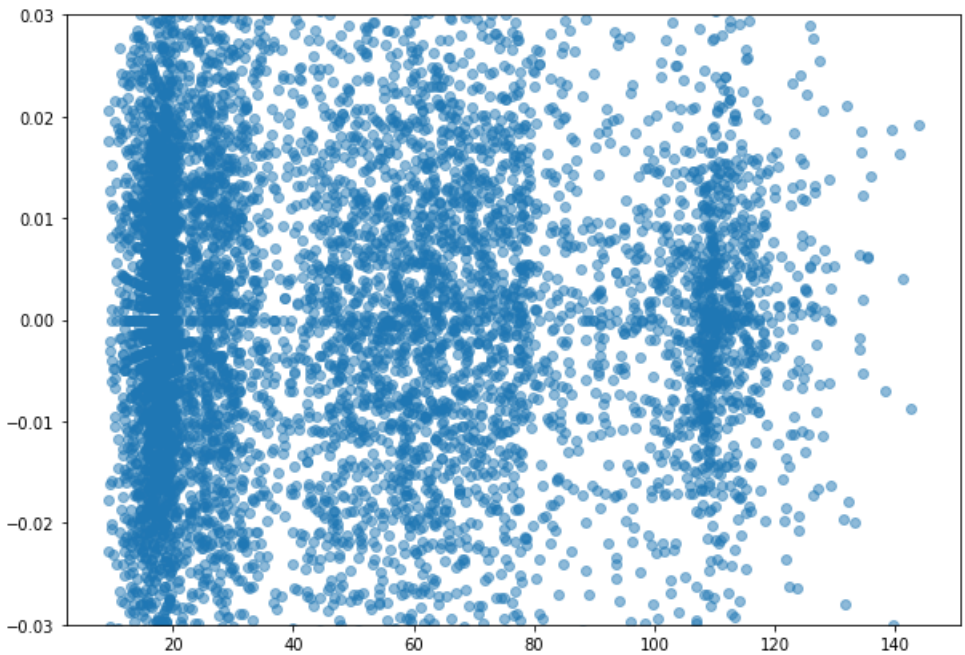
Out[10]:
```
<matplotlib.collections.PathCollection at 0x7fd7c17b1fd0>
```



Şimdi ise yukarıdaki grafikte daha dar bir alana göz atalım

In [11]:
```
scatter(data["Europe Brent Spot Price FOB (Dollars per Barrel)"],data.pct,alpha=0.5)
ylim(-0.03,0.03)
```

Out[11]:
```
(-0.03, 0.03)
```



Yukarıda dikkat ettiminiz mi bilmiyorum ama özellikle belirli fiyat seviyelerinde bir kümelenme var. Mesela 20, $40-80$ ve 110$

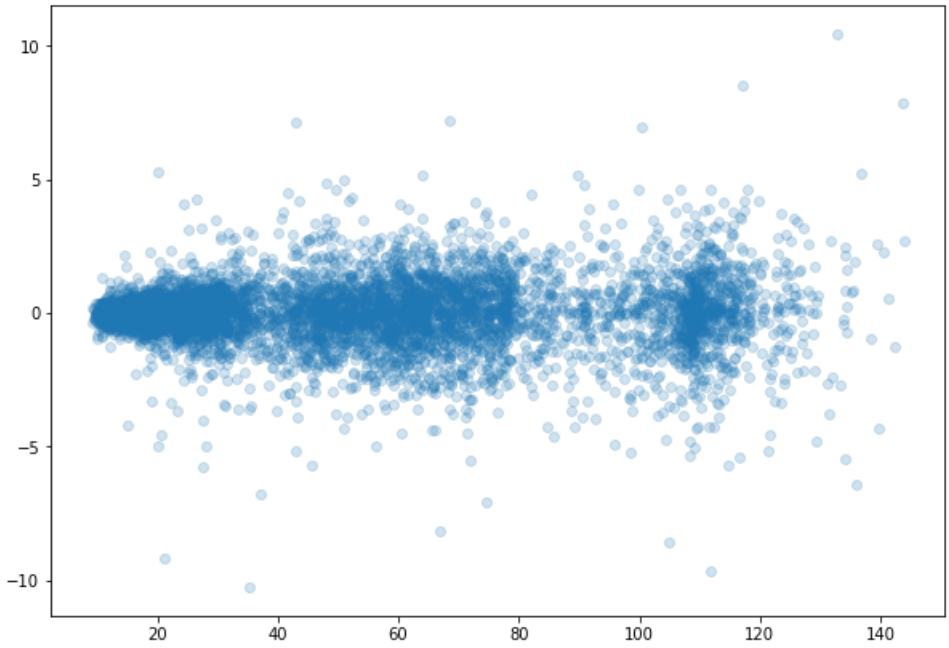
Veri satırımızda % değişimler için pct sütununu açmıştık. Şimdi de mutlak farklar için "dif" sütununu açalım

In [12]:
```
data["dif"]=data["Europe Brent Spot Price FOB (Dollars per Barrel)"].diff()
```

Değişik fiyat seviylerindeki oynamanın tam sayı değerlerini kıyasladığımızda da benzer bir kesikli gruplama görüyoruz

In [13]:
```
scatter(data["Europe Brent Spot Price FOB (Dollars per Barrel)"],data["dif"],a
lpha=0.2)
```

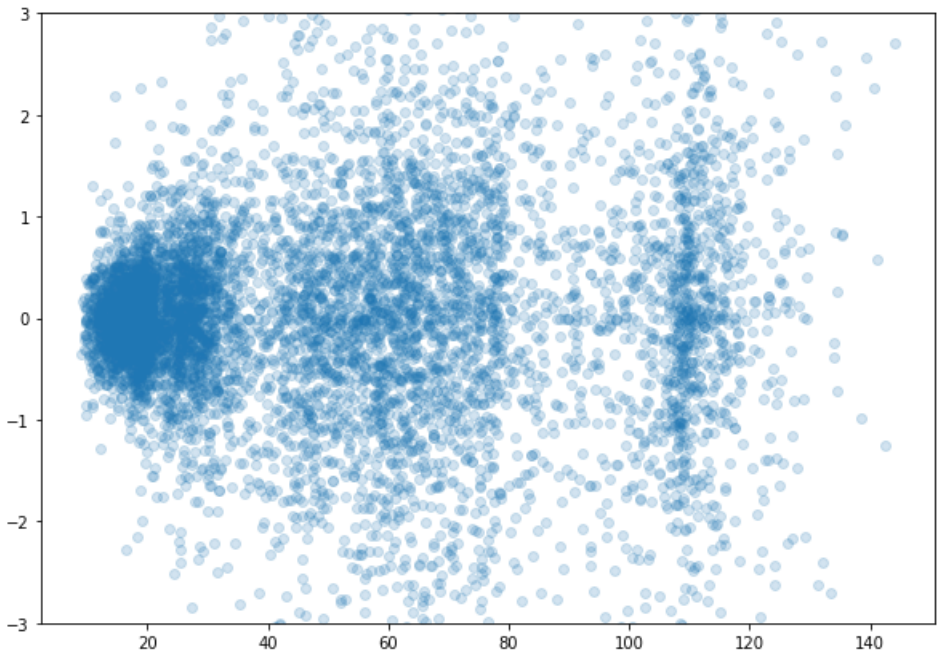
Out[13]:
```
<matplotlib.collections.PathCollection at 0x7fd7c1701e80>
```



Bir öncekindeki gibi biraz daha yakından bakalım

In [14]:
```
scatter(data["Europe Brent Spot Price FOB (Dollars per Barrel)"],data["dif"],a
lpha=0.2)
ylim(-3,3)
```

Out[14]: (-3, 3)



Bölüm 3 - Monte Carlo ile 2020 ilk dönemindeki değişimi tahmin

2020 yılının ilk 3 ayının verileri **data_s** değişkenindeydi. Şimdi bu dönemdeki yüzde değişimleri (pct_change) ile hesaplayarak, sayı olmayan sonuçları çıkaralım (dropna). Hesaplanan verileri de bir listeye çevirerek (.values) ile **lst** değişkenine alalım.

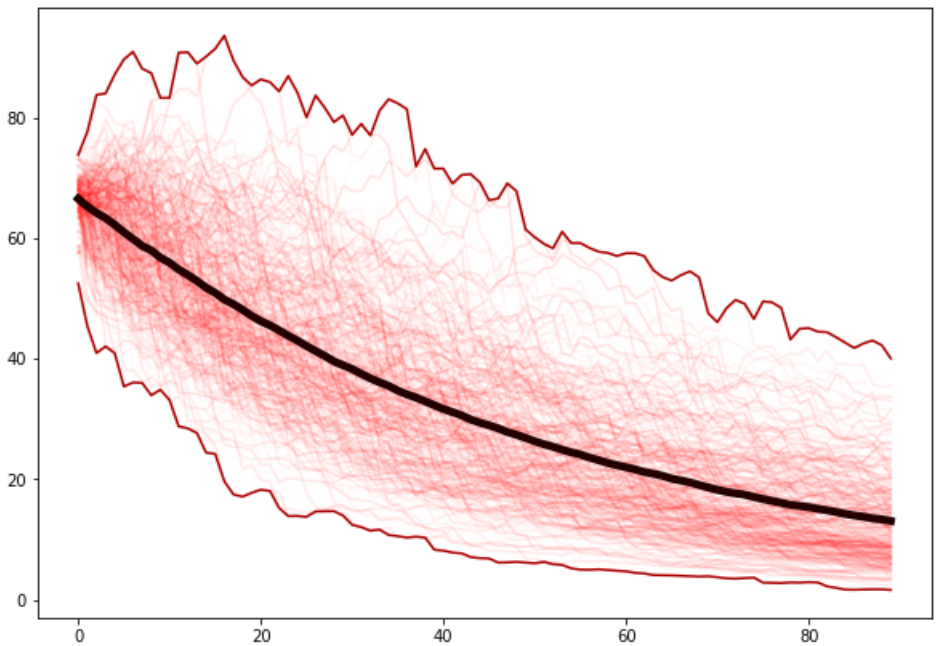
In [15]:
```
#ilk çeyrekte fiyat değişim oranı
lst=data_s["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_change().dr
opna().values
```

Ben bir kaç seferde aşağıdaki kodu yazdığımdan kısaca ne yaptığını özetleyeyim

- Önce bir başlangıç petrol fiyatı ile başlar
- Daha önceden 2020 1.çeyrekteki fiyat değişimlerini lst değişkenine atmıştık. Bu değişimlerden rastgele oranları alarak arka arkaya petrol fiyatı ile çarpar
- **days** ile ileri doğru 90 gün için
- **numsim** ile 300 defa tekrarlayarak
- En yüksek değerleri pmax, en düşük değerleri pmin, ortalamaları da pmid'e alır
- Ardından da grafiği çizdiriri

In [16]:
```
pricematrix=[]
startprice=67.7
days=90
numsim=300
for ax in np.arange(numsim):
 sprice=startprice
 price=[]
 lsta=random.choice(lst,size=days)
 for ai in np.arange(days):
    sprice=sprice*(1+lsta[ai])
    price.append(sprice)
 pricematrix.append(price[:])

pmid=[]
pmin=[]
pmax=[]
npm=np.array(pricematrix)
for ax in np.arange(days):
 pmax.append(np.amax(npm[0:numsim,ax]))
 pmin.append(np.amin(npm[0:numsim,ax]))
 pmid.append(np.average(npm[0:numsim,ax]))


for ax in np.arange(numsim):
 plot(npm[ax,:], alpha=0.1*ax/numsim, color="red");


plot(pmax, color="#AA0000")
plot(pmin, color="#AA0000")
plot(pmid, color="#220000", linewidth=5)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7fd7c6375550>]
```



Görüldüğü üzere 2020 1.çeyrekteki değişim oranlarını ne kadar rastgele hale getirirsek getirelim en iyi ihtimalle bile fiyat düşüşü kaçınılmaz oluyor. Gerçekten de çok kötü bir dönem. Fiyat hatta spot olarak 20 dolar/varil in altına iniyor ki, bu değere 8-10 dolar/varil ekleyerek ekranda gördüğümüz değerlere de ulaşabilirsiniz

Bölüm 4 - Monte Carlo fonksiyonu

Şimdi de bu yaptığımız çalışmayı daha sistematik hale getirmek için bir fonksiyona dönüştürelim. Fonksiyonumuzda

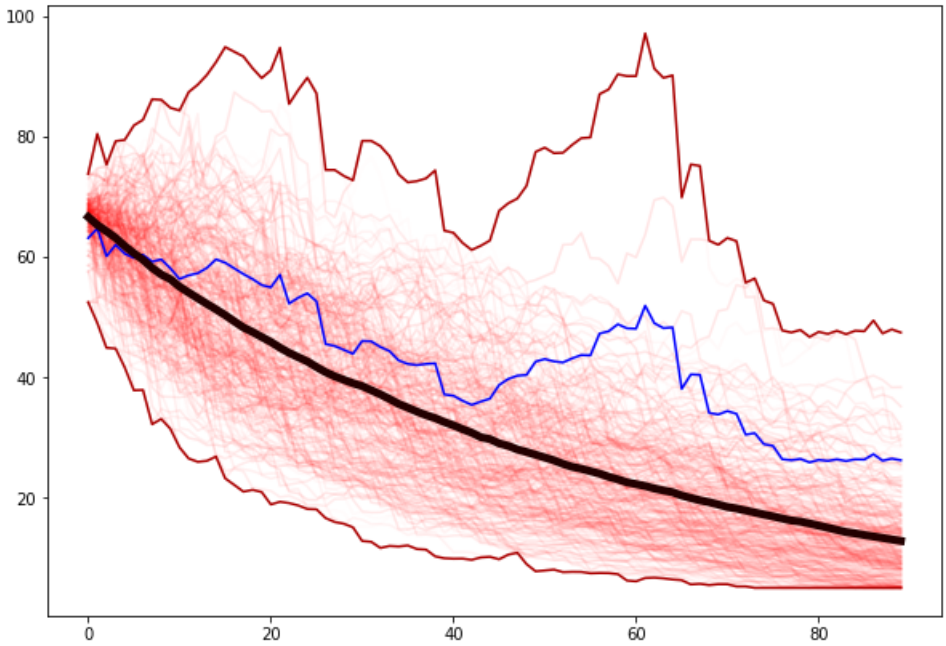
- sp: Başlangıç fiyatı
- days: ileri doğru kaç gün simulasyonu ilerletmek istediğimiz
- lstx: petrol fiyat değişimi kümesi (benzer dönemlerden örneklem seçmek için), program bu kümeden rastgele oranlar seçer
- numsim: bu simulasyonu kaç defa çalıştırmak istediğimiz

In [17]:
```
def montecarlo(sp, days, lstx, numsim=300):
 pricematrix=[]
 days=days
 for ax in np.arange(numsim):
  sprice=sp
  price=[]
  lsta=random.choice(lstx,size=days)
  for ai in np.arange(days):
    sprice=sprice*(1+lsta[ai])
    if (sprice<5): sprice=5
    price.append(sprice)
  pricematrix.append(price[:])

  pmid=[]
  pmin=[]
  pmax=[]
  pext=[]
  npm=np.array(pricematrix)

 for ax in np.arange(days):
  pmax.append(np.amax(npm[0:numsim,ax]))
  pmin.append(np.amin(npm[0:numsim,ax]))
  pmid.append(np.average(npm[0:numsim,ax]))
  pext.append(((np.amax(npm[0:numsim,ax])+np.amin(npm[0:numsim,ax]))/2))

 for ax in np.arange(numsim):
  plot(npm[ax,:], alpha=0.1*ax/numsim, color="red");

 plot(pmax, color="#AA0000")
 plot(pmin, color="#AA0000")
 plot(pext, color="blue")
 plot(pmid, color="#220000", linewidth=5)
 print ("Ortalama=",pmid[days-1], " Uçların ortalaması=",pext[days-1])
 return pmid[:]
 #ylim(0,sp+50)
```

Fonkiyonumuzu test edelim, 67.7 dolardan başlarsak, 90 gün sonra simulasyonumuzdaki fiyat aralığı ne olur?

In [18]:
```
montecarlo(67.7, 90, lst);
```

```
Ortalama= 12.805560977194332  Uçların ortalaması= 26.214292830730653
```



Bölüm 5 - 2020 yılı İkinci çeyrek için fiyat tahmini

Geçtiğimiz çeyrekteki değişimler ile petrol fiyatlarını Ocak ayından aldığımızda ne kadar simulasyon yaparsak yapalım fiyatların düştüğünü gördük. Burada en önemli sorun "şişman kuyruklar". Evet petrol fiyat değişimleri ortalama dağılıyor gibi gözükse de, 50 seçimden birinde %10 düşüş büyük bir yıkım oluyor. Sonuçlarda bu rahatlıkla görülüyor.

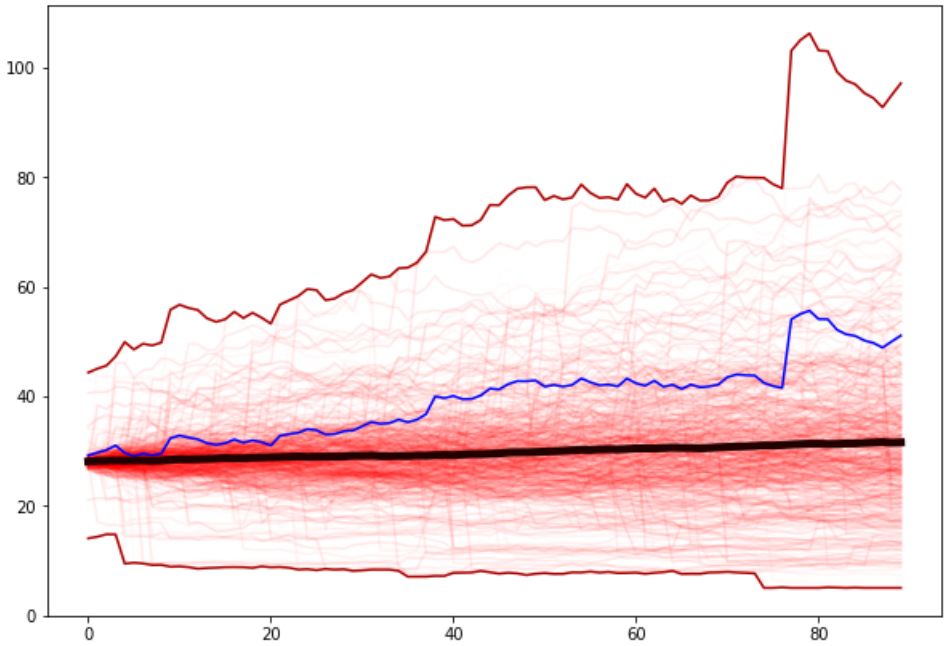
Şimdi de 2020 yılından önceki yıllardaki 4,5,6.aylardaki değişim oranları ile MonteCarlo simulasyonumuzu çalıştıralım. Aşağıda değişik dönemler için de filtreleme kodları # ile verilmiştir

In [19]:
```
# 2.çeyrek
data_tmp=data[(data["Date"].dt.year<2020) & (data["Date"].dt.month>3) & (data
["Date"].dt.month<7) ]
# 3.çeyrek
# data_tmp=data[(data["Date"].dt.year<2020) & (data["Date"].dt.month>6) & (dat
a["Date"].dt.month<10) ]
# 4. çeyrek
# data_tmp=data[(data["Date"].dt.year>2010) & (data["Date"].dt.month>9) ]

# data_tmp=data[(data["Date"].dt.year>2016) & (data["Date"].dt.month>0) ]
#data_tmp=data[(data["Date"].dt.month>3)  &  (data["Date"].dt.month<5) ]
lst_tmp=data_tmp["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_chang
e().dropna().values
```

28 dolardan başlatarak, 90 gün ileri doğru, 500 defa simulasyon yapalım, ortalama fiyatları da rslt listesine kopyalayalım

In [20]:
```
rslt=montecarlo(28, 90 , lst_tmp[:], 500);
```

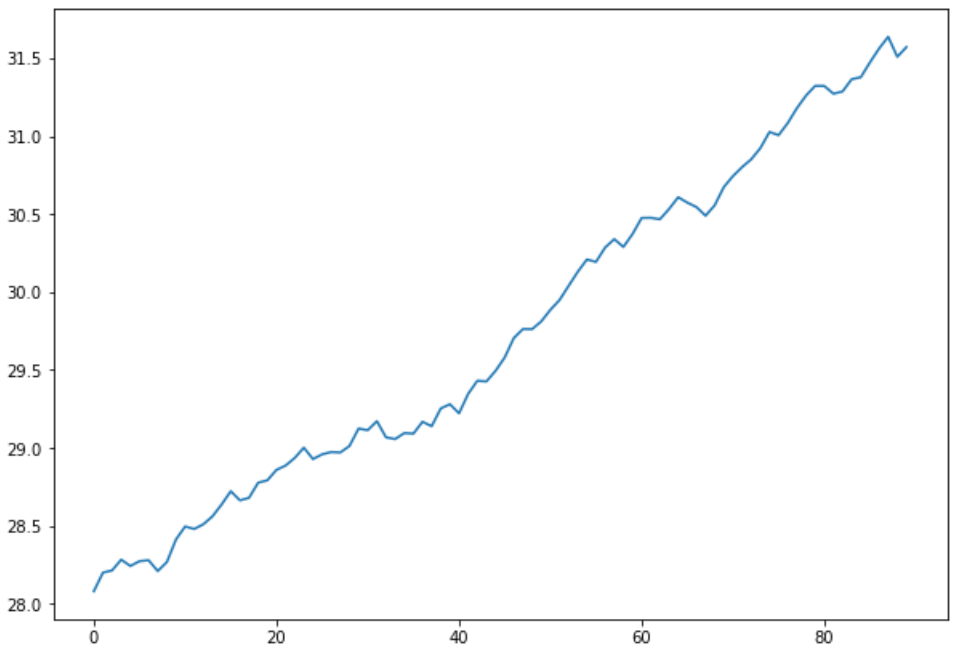
```
Ortalama= 31.57092987151359  Uçların ortalaması= 51.08134502106963
```



Peki ortalama olarak fiyatlar nereye gidiyormuş bir de onu görelim. Simulasyonu spot fiyatlar üzerinden yaptık. 13'ünde Brent spot 20 dolardan kapanmış, ama o gün gelecek fiyatı 30 dolar civarında. Yani 10 dolar da spottan gelecek kontrata bir contango olduğu için ekleme yapabiliriz.

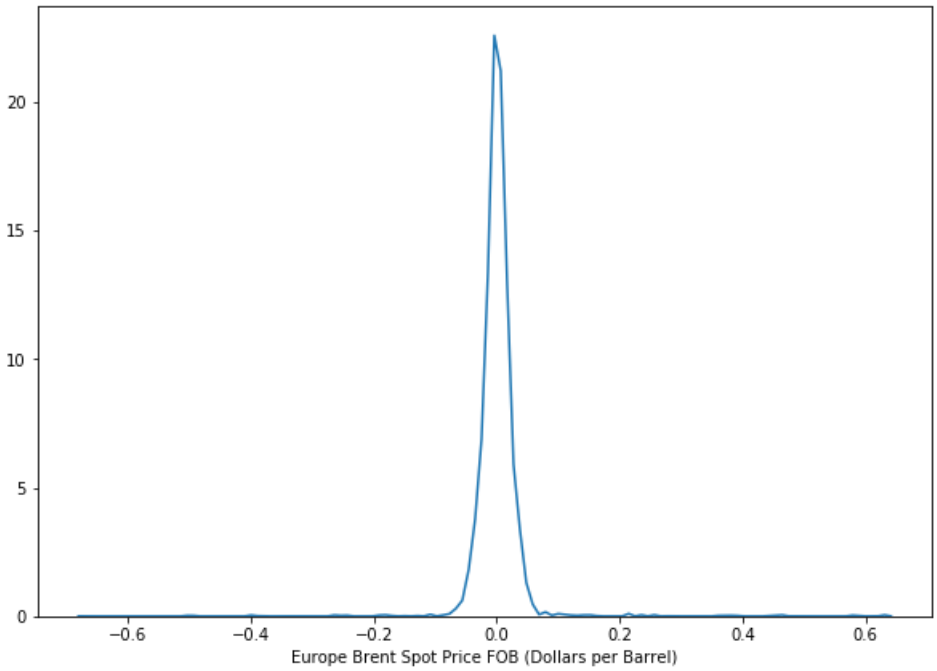
In [21]:
```
plot(rslt)
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x7fd7c01a6240>]
```



In [22]:
```
sb.distplot(data_tmp["Europe Brent Spot Price FOB (Dollars per Barrel)"].pct_c
hange().dropna(), hist=False, rug=False, hist_kws={"range": [0,1]})
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd7bb55e080>
```



Bölüm 6 - Fiyat seviyelerine özel yüzde değişim ve seviye değişimi var ise

Şimdi ise bir başka teoriye bakalım. Petrol fiyatı bir seviyeye geldiğinde -mesela 20 dolar'a-, değişim oranı o seviyedeki değişim oranlarına benzeşiyor olabilir. Bu sebeple bu sefer biraz daha farklı bir yöntem uygulayacağız.

Önce değişim ve farkları bir satır yukarı kaydıralım. Sebebi de basit petrol 20 dolardan 22 dolara çıktı ise %10 artış oranı olması fiyatın 22 değil 20 dolarda olmasına bağlı. Ama fonksiyon bunu değişen miktar yani 22 ile aynı satıra basıyor.

In [23]:
```
data["pct"]=data["pct"].shift(-1)
data["dif"]=data["dif"].shift(-1)
```

Sürekli tırnak içine uzun uzun "Europe " vs yazmaktan usandığım için işime yarayan verileri **data2** isimli ayrı bir veri noktasına alıyorum ve uzun isimli sütunun adını **price** olarak değiştiriyorum

In [24]:
```
data2=data[["Europe Brent Spot Price FOB (Dollars per Barrel)","pct","dif"]]
```

In [25]:
```
data2=data2.rename(columns={"Europe Brent Spot Price FOB (Dollars per Barre
l)":"price"})
```

Verimize bir bakalım

In [26]:
```
data2.head()
```

Out[26]:

	price	pct	dif
0	18.63	-0.009662	-0.18
1	18.45	0.005420	0.10
2	18.55	0.002695	0.05
3	18.60	0.001613	0.03
4	18.63	-0.001610	-0.03

Şimdi de verimizi en küçükten en büyüğe sıralayalım

In [27]:
```
data2=data2.sort_values(by=['price'])
```

In [28]:
```
data2.head()
```

Out[28]:

	price	pct	dif
2937	9.10	0.017582	0.16
2938	9.26	0.020518	0.19
2944	9.45	0.040212	0.38
2939	9.45	0.012698	0.12
2936	9.46	-0.038055	-0.36

Kod nasıl çalışıyor? Mesela başladığımız fiyat 60 dolar olsun. Fiyat hareketlerinin 50-70 dolar arasındaki kümesinden rastgele bir artış oranını aşağıdaki şekilde alıyoruz. Yani burada her fiyat seviyesinde o seviyeye özgü bir değişim oranı varmış gibi yaklaşıyoruz

In [29]:
```
targetprice=60
100*random.choice(data2[(data2.price>(targetprice-10))  & (data2.price<(target
price+10))].pct.values)
```

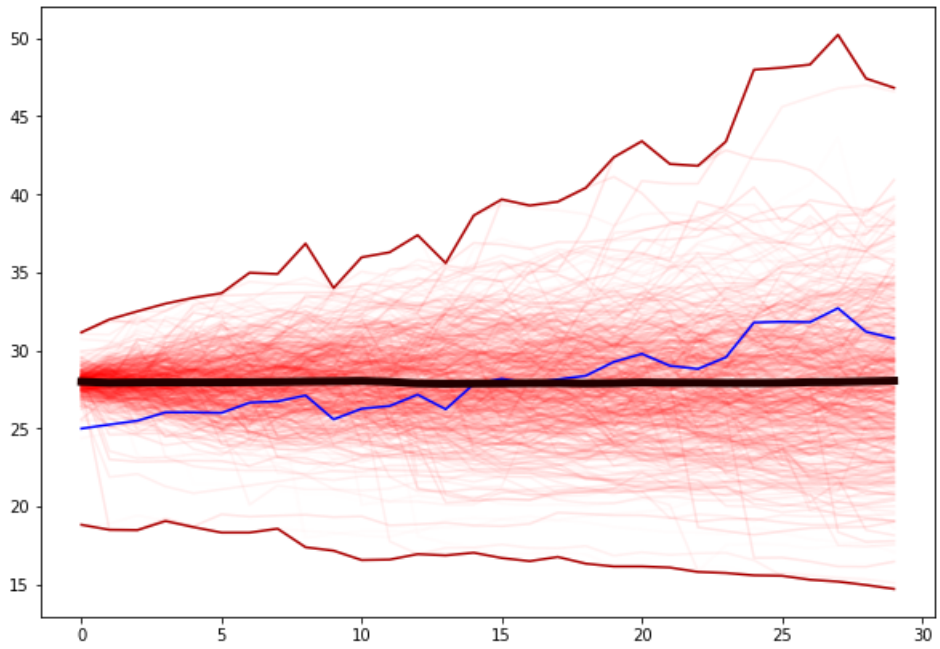
Out[29]:
```
4.451904831411468
```

Şimdi yukarıdaki MonteCarlo kodumuzu önce her fiyat seviyesinde benzer bir mutlak fiyat hareketi varmış gibi çalıştıralım. Yani yüzde değişim değil "dif" yani farklara bakalım

In [30]:
```
startprice=28
days=30
numsim=500
pricematrix=[]
days=days
for ax in np.arange(numsim):
    sprice=startprice
    price=[]
    for ai in np.arange(days):
        arr=data2[(data2.price>(sprice-5))  & (data2.price<(sprice+5))].dropna
()
        change=random.choice(arr["dif"].values)
#        change=average(arr["diff"].values)
        sprice=sprice+change
        price.append(sprice)
    pricematrix.append(price[:])
```

In [31]:
```
pmid=[]
pmin=[]
pmax=[]
pext=[]
npm=np.array(pricematrix)

for ax in np.arange(days):
  pmax.append(np.amax(npm[0:numsim,ax]))
  pmin.append(np.amin(npm[0:numsim,ax]))
  pmid.append(np.average(npm[0:numsim,ax]))
  pext.append(((np.amax(npm[0:numsim,ax])+np.amin(npm[0:numsim,ax]))/2))

for ax in np.arange(numsim):
  plot(npm[ax,:], alpha=0.1*ax/numsim, color="red");

plot(pmax, color="#AA0000")
plot(pmin, color="#AA0000")
plot(pext, color="blue")
plot(pmid, color="#220000", linewidth=5)
print ("Ortalama=",pmid[days-1], " Uçların ortalaması=",pext[days-1])
```

```
Ortalama= 28.047420000000002  Uçların ortalaması= 30.765000000000015
```



Görüldüğü üzere böyle bir durumda fiyat değişimlerinde ortalamada daha farklı sonuçlar elde ediliyor.

Peki bir de % değişimlere baksak neler oluyor ona bakalım. Başlangıç fiyatımızı 28 dolardan ileri doğru 30 gün 500 defa simulasyonumuzu tekrarlayalım

In [32]:
```
startprice=28
days=30
numsim=500
pricematrix=[]
days=days
for ax in np.arange(numsim):
    sprice=startprice
    price=[]
    for ai in np.arange(days):
        arr=data2[(data2.price>(sprice-5))  & (data2.price<(sprice+5))].dropna
()
        change=random.choice(arr["pct"].values)
#        change=average(arr["diff"].values)
        sprice=sprice*(1+change)
        price.append(sprice)
    pricematrix.append(price[:])

pmid=[]
pmin=[]
pmax=[]
pext=[]
npm=np.array(pricematrix)

for ax in np.arange(days):
  pmax.append(np.amax(npm[0:numsim,ax]))
  pmin.append(np.amin(npm[0:numsim,ax]))
  pmid.append(np.average(npm[0:numsim,ax]))
  pext.append(((np.amax(npm[0:numsim,ax])+np.amin(npm[0:numsim,ax]))/2))

for ax in np.arange(numsim):
  plot(npm[ax,:], alpha=0.1*ax/numsim, color="red");

plot(pmax, color="#AA0000")
plot(pmin, color="#AA0000")
plot(pext, color="blue")
plot(pmid, color="#220000", linewidth=5)
print ("Ortalama=",pmid[days-1], " Uçların ortalaması=",pext[days-1])
```

```
Ortalama= 27.810240205886142  Uçların ortalaması= 31.714360823592216
```