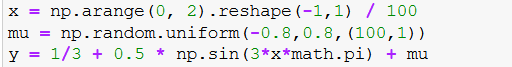
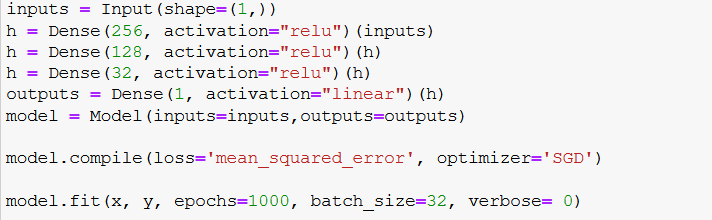
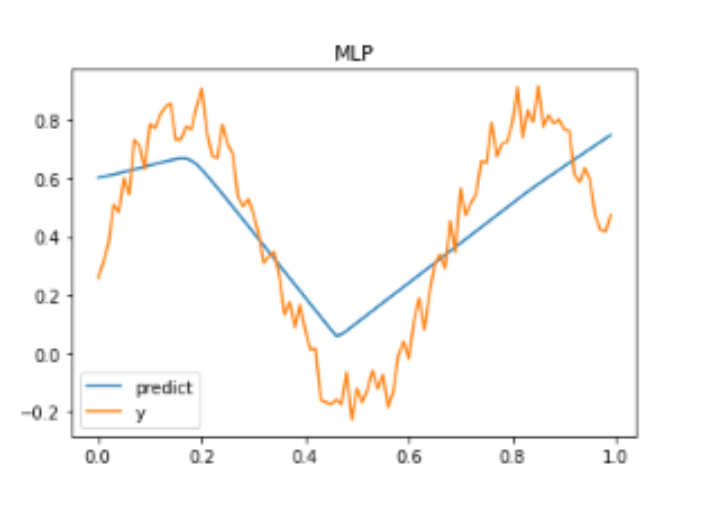
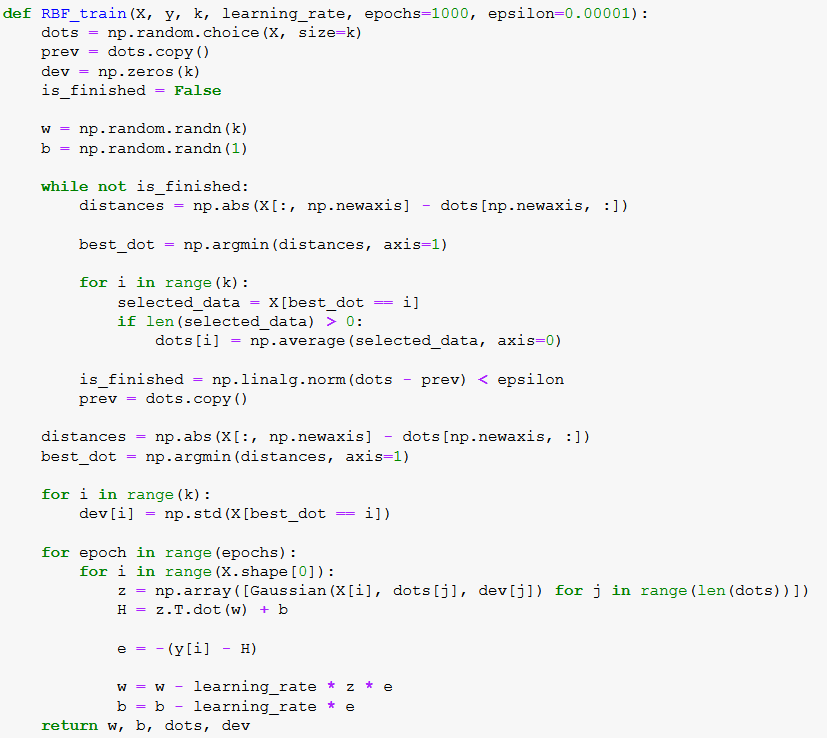
**: MLP**  
به این شکل Xو yخود را آماده می کنیم.

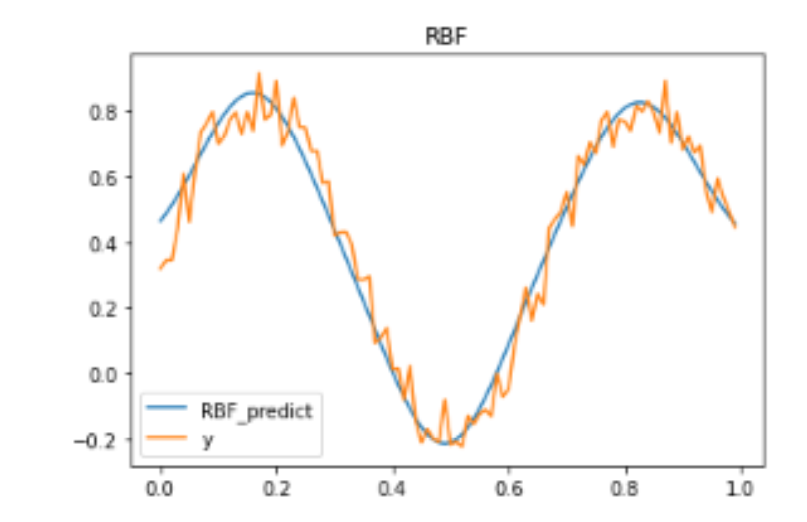
  
مدل را اینگونه تعریف و trainمی کنیم.

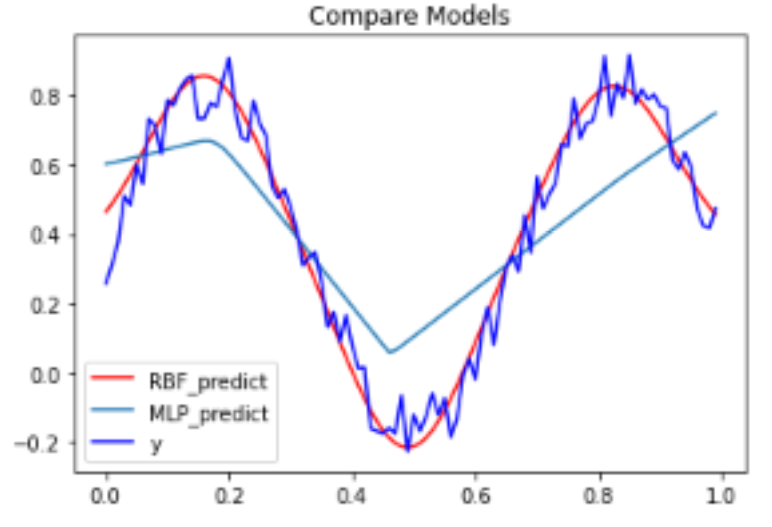
  
نتیجه به دست آمده از مدل :

  
**: RBF**  
ابتدا تابع گاوسی را می نویسیم تا در ادامه از آن استفاده بکنیم

  
بخش اصلی الگوریتم در تابع RBF\_trainقرار داده شده است.

  
ابتدا kتا از داده های موجود را به عنوان نقطه همگرایی به صورت تصادفی انتخاب می کنیم.  
سپس وزن ها ها را به صورت رندوم تعریف می کنیم.  
در نهایت نزدیک ترین نقطه همگرایی را برای هر کدام از داده ها را پیدا می کنیم و در آخر مکان هر نقطه همگرایی را به میانگین داده های assignشده به آن نقطه تغییر می دهیم. این کار را تا جایی انجام می دهیم که مقدار حرکت نقاط همگرایی نسبت به حالت قبلشان کمتر از epsilonشود.  
در ادامه تابع RBF\_trainبعد از آن که محل نقاط همگرایی setشدند به trainشبکه می پردازیم.  
با استفاده از تابع predictنیز عملکرد مدل خود را می سنجیم.  
نتیجه به دست آمده از مدل :

  
مقایسه نتایج :

  
نتایج RBFخیلی بهتر از MLPاست.  
**مقایسه MLPو : RBF**  
RBFسریعتر میباشد  
RBFراحت تر می تواند با دیتا شامل noiseکار بکند   
MLPبر اساس ضرب داخلی کار میکند اما RBFبر اساس فاصله اقلیدسی  
اگر در MLPچند نورون را حذف کنیم عملکرد دچار اختلال بزرگ نمی شود (حتی مقالات زیادی در مورد هرس مدل های MLPوجود دارد تا بتوان از آن ها در گوشی مبایل هم استفاده کرد) ولی اگر یکی از نقطه همگرایی که در کل kتا هستند را حذف کنیم کل مدل به هم می ریزد.  
MLPبه ترتیب ورودی ها وابسته است  
**ترکیب MLPبا : RBF**  
می توانیم با استفاده از MLPداده های خود را به شکل دلخواهمان در بیاوریم و سپس داده های جدید را به مدل RBFبه عنوان ورودی می دهیم

