Supplementary Material

A machine-learning framework for predicting high confident copy number variants

Taifu Wang[1*], Jinghua Sun[1,2*], Xiuqing Zhang[1,2,3], Wen-Jing Wang[1#] and Qing Zhou[1#]

[1]BGI-Shenzhen, Shenzhen 518083, China.

[2]College of Life Sciences, University of Chinese Academy of Sciences, Beijing 100049，China

[3]Guangdong Enterprise Key Laboratory of Human Disease Genomics, Beishan Industrial Zone, Shenzhen, 518083, China.

Supplementary Methods

CNV Calling processes

For the WGS data, we aligned it to the hg19 reference with bwa(bwa-0.7.16a) 'mem' command to generate the BAM file. Then, the BAM file was analyzed by multiple tools for detecting CNVs. The commands and options used for the 5 software used in this study are described below (Taking HG002 for example).

1. Lumpy

For running Lumpy, we implanted SpeedSeq workflow. We firstly extracted discordant paired read alignments and split read alignments with samtools and the extractSplitReads_BwaMem command. The 'lumpy' command was executed with the options (e.g. HG002):

-pe bam_file: HG002.discordants.bam, histo_file: HG002.lib1.x4.histo, mean: 569.0479672525416, stdev: 156.96862863277676, read_length: 148, min_non_overlap: 148, discordant_z: 5, back_distance: 10, weight: 1, id: HG002, min_mapping_threshold: 20, read_group: HG002

-sr bam_file: HG002.splitters.bam, back_distance: 10, min_mapping_threshold: 20, weight:1, id: HG002, min_clip: 20.

2. manta

For running Manta, we ran 'configManta.py' script on the input bam file and hg19 reference file, then, the 'runWorkflow.py' script was used to execute the entire workflow. The 'manta' was executed with the following commands:

1) $Bin/configManta.py --bam HG002.bam --referenceFasta hg19.fa --runDir ./

2) $Bin/runWorkflow.py -j 8

3. delly

For delly, we ran 'delly call' command on the input bam file and hg19 reference file to get the BCF file, then bcftools was executed for converting BCF to VCF.

commands used:

1) $Bin/delly call -t ALL -g hg19.fa -o HG002.delly.bcf HG002.bam

2) $Bin/bcftools view HG002.delly.bcf > HG002.delly.vcf

4. pindel

We created a configure file for each chromosome, in which the input bam file and the mean insert size were specified. Next, the 'pindel' command was executed for each chromosome using the hg19 reference and configure file. Finally, pindel2vcf was executed for converting output files to a VCF file.

commands as follows:

1) $Bin/pindel -f hg19.fa -i HG002. Configure_file -c chr1 -T 16 -x 2 -M 3 -v 100 -d 30 -E 0.92 -w 10 -o HG002.chr1

2) $Bin/pindel2vcf -r hg19.fa -R hg19 -P HG002.chr1 -v HG002.chr1.sv.vcf

5. Breakdancer

For breakdancer, we executed 'bam2cfg.pl' script to create configure files. Then, breakdancer-max was executed for detecting CNVs.

commands used:

1) $Bin/ bam2cfg.pl -v 20 HG002.bam > HG002.SV.cfg

2) $Bin/ breakdancer_max -y 30 -x 1000 -r 2 -m 10000000 HG002.SV.cfg > HG002.SV.ctx

6. CNV-JACG

To run CNV-JACG, we split vcf files into two separate files, which contain duplications and deletions respectively, and then executed 'CNV-JACG.pl' script for CNV calling.

1) perl $Bin/CNV-JACG.pl -p HG002.DEL.vcf -b HG002.bam -r hg19.fasta -o DEL_OUT

2) perl $Bin/CNV-JACG.pl -p HG002.DUP.vcf -b HG002.bam -r hg19.fasta -o DUP_OUT

7. MetaSV

MetaSV is An accurate and integrative structural-variant caller. Since MetaSV currently does not support reading in Delly's output, we only integrate the results of the four software (Lumpy, Manta, Pindel, and breakdancer).

commands used:

```
run_metasv.py --reference hg19.fasta \
            --boost_ins \
            --pindel_vcf pindel.HG002.vcf \
            --breakdancer_vcf breakdancer. HG002.vcf \
            --manta_vcf manta.HG002.vcf \
            --lumpy_vcf Lumpy.HG002.vcf \
            --sample HG002 \
            --bam HG002.rmdup.bam \
            --spades SPAdes/spades.py \
            --age AGE/age_align \
            --num_threads 5 \
            --workdir HG002 \
            --outdir HG002/HG002_out \
            --min_support_ins 2 --max_ins_intervals 500000 \
            --isize_mean 569 --isize_sd 95
```

8. SURVIVOR

SURVIVOR was used to merge the filtered results by CNV-P, CNV- JACG and hard cutoff method, setting 80% as the cut-off of rate of overlapped region.

commands used:

```
$Bin/SURVIVOR merge sample.sv.list 0.8 1 1 0 0 100 HG002.mer.vcf
```

Supplementary Figures

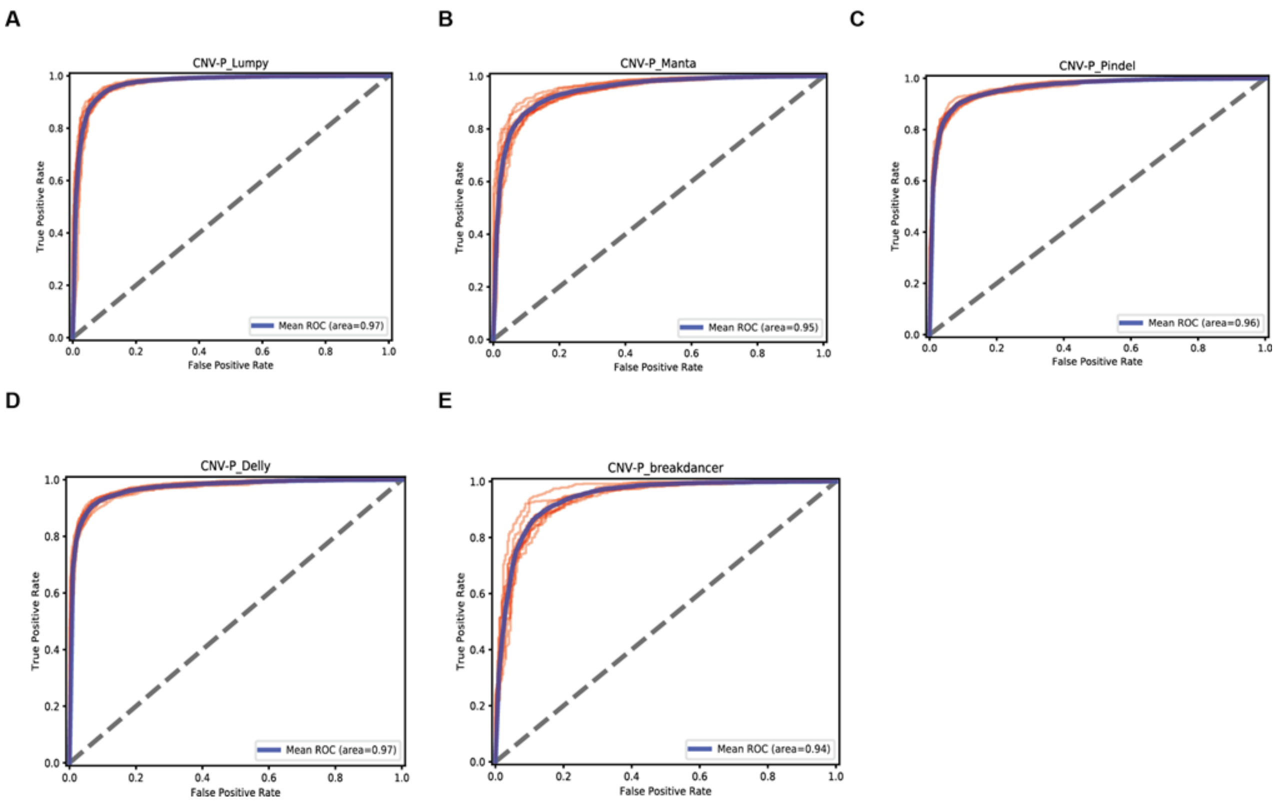


Figure S1. 10-fold Cross validation for CNV-P. For each classifier, we performed cross-validation of the training set in ten folds. A) Lumpy; B) Manta; C) Pindel; D) Delly; E) breakdancer. The red line showed the ROC for each cross-validation and blue lines showed the mean ROC curve.

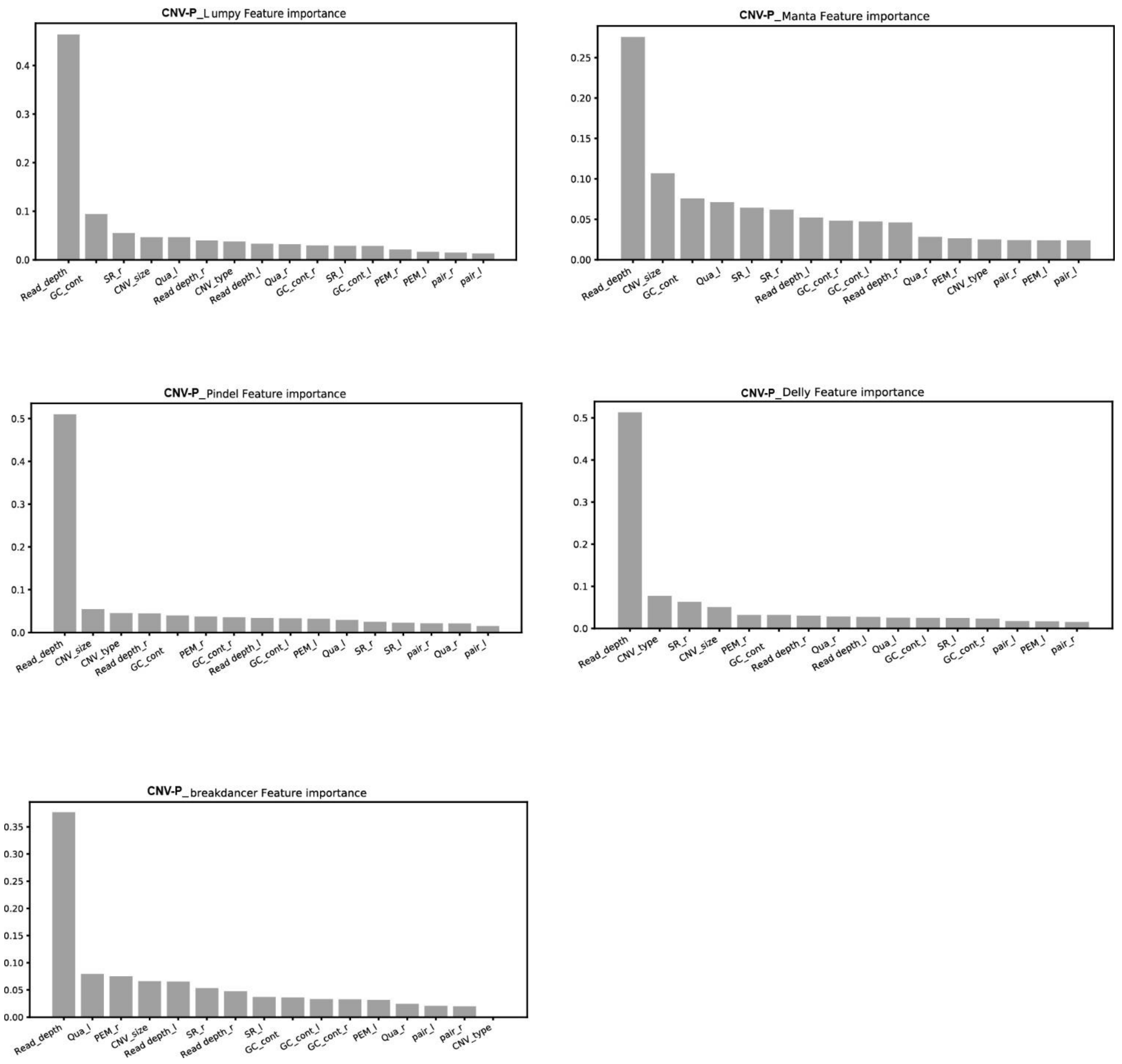


Figure S2. Importance of features in CNV-P. We trained CNV-P classifier for each CNV software using Random Forest (RF). The features were ranked by relative importance.

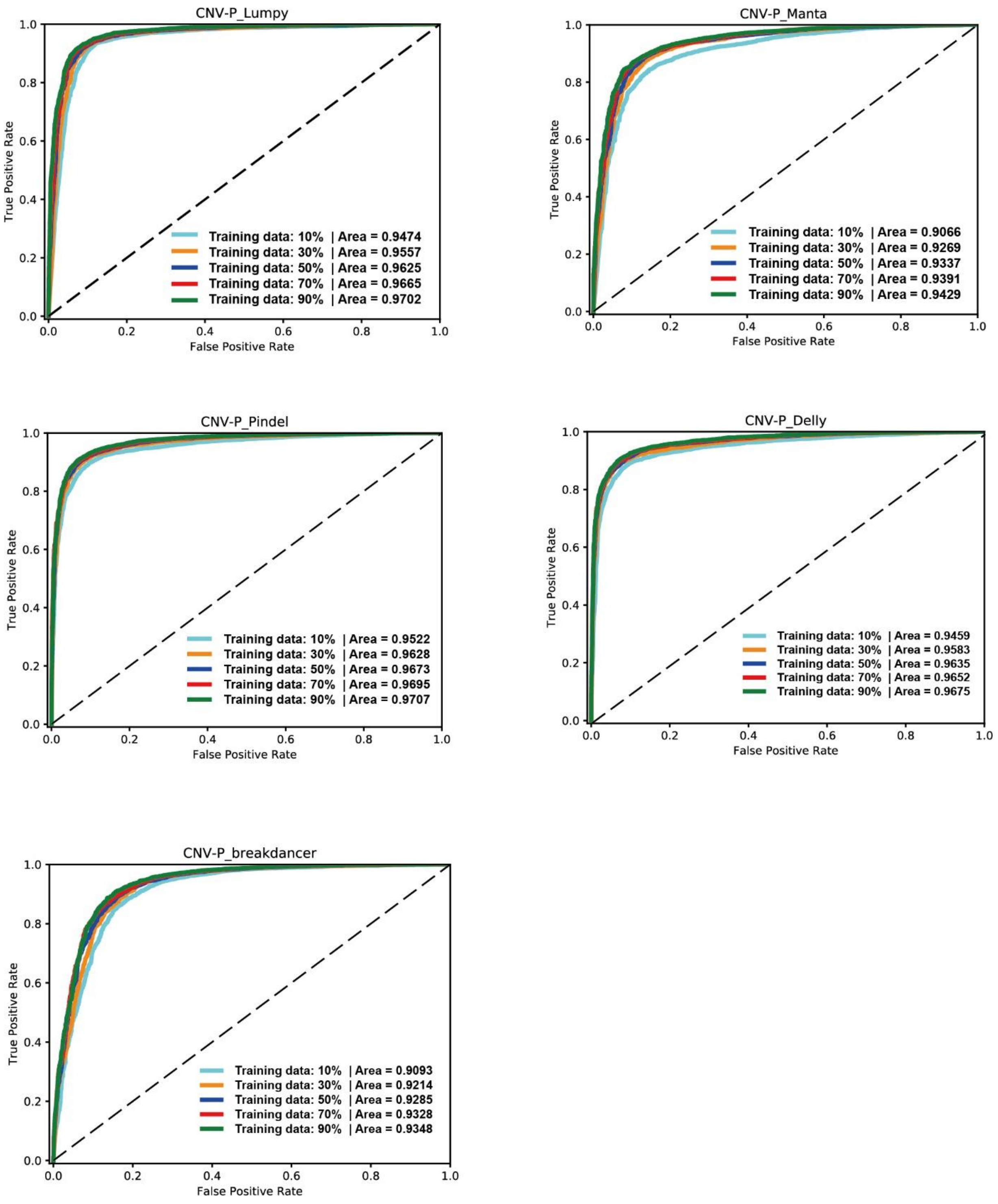
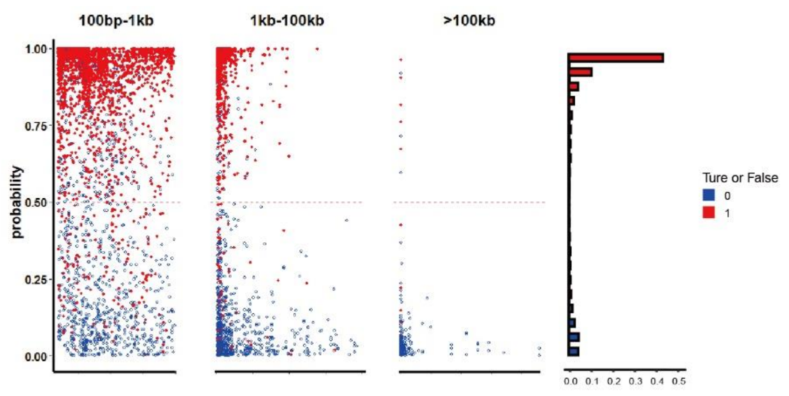
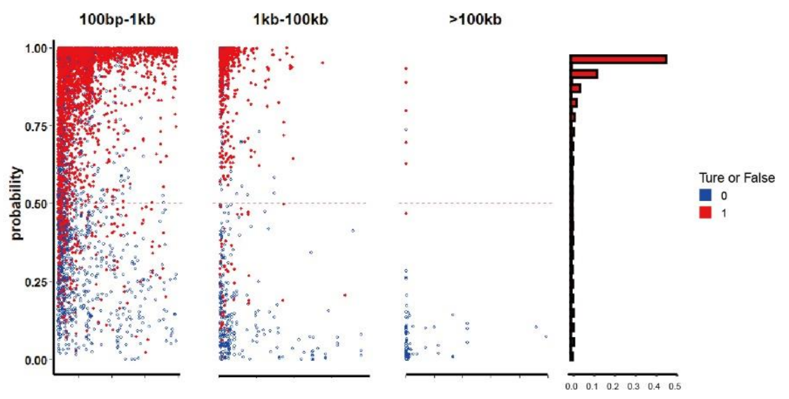


Figure S3. Classifier improves with increase of training data. To evaluate the robustness of each CNV-P, we trained each CNV-P on varying proportions of training data (from 10% to 90% in increments of 20%).

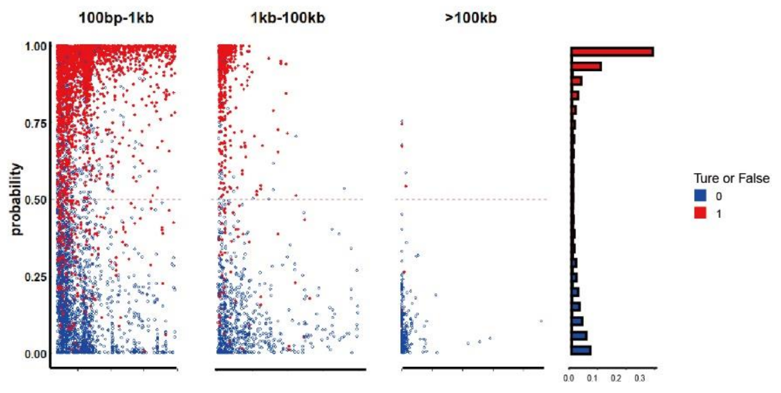
A) CNV-P_Lumpy



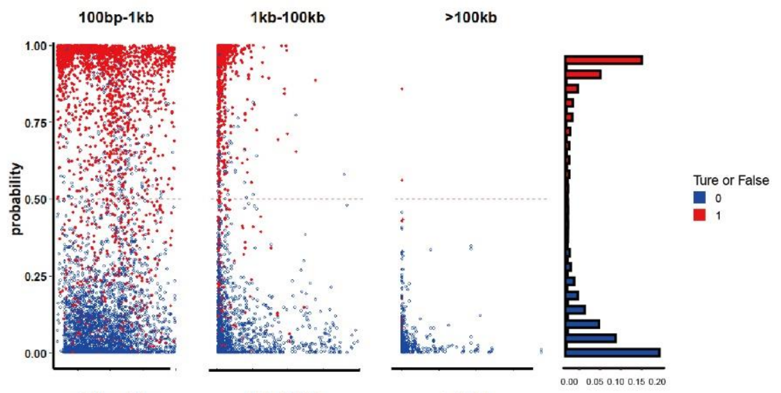
B) CNV-P_Manta



C) CNV-P_Pindel



D) CNV-P_Delly



E) CNV-P_breakdancer

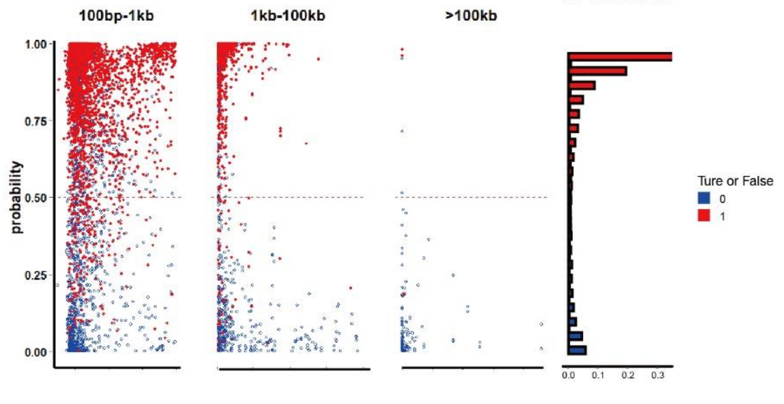
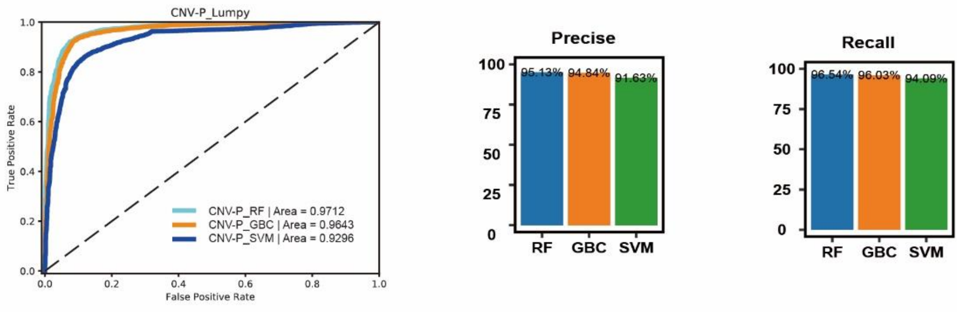
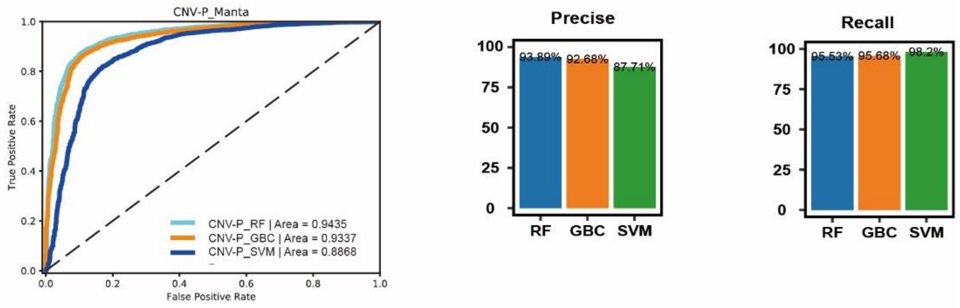


Figure S4. Distribution of predicted probability scores of CNVs at different size ranges. The distribution of all the probability scores, predicted by CNV-P, is shown on the left, and the frequency distribution across 20 bins is shown on the right. A) Lumpy; B) Manta; C) Pindel; D) Delly; E) breakdancer

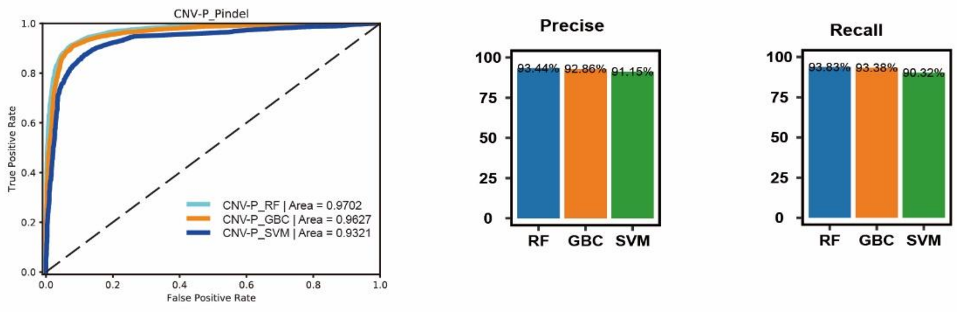
A) CNV-P_Lumpy



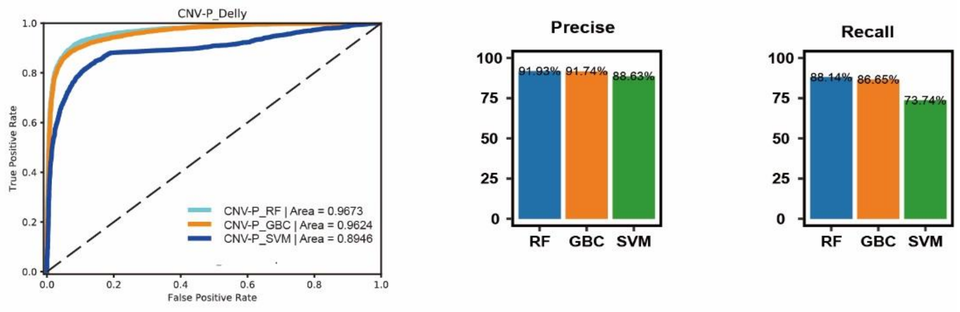
B) CNV-P_Manta



C) CNV-P_Pindel



D) CNV-P_Delly



E) CNV-P_breakdancer

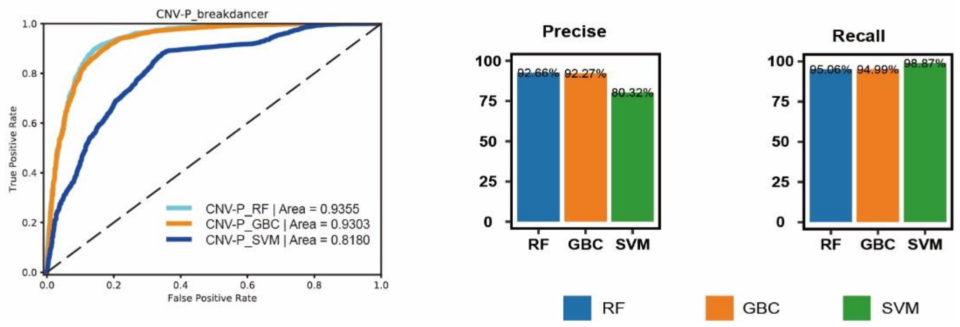


Figure S5. Comparison of performance using three classifiers supported by CNV-P. For each CNV-P classifier A) Lumpy; B) Manta; C) Pindel; D) Delly; E) breakdancer, ROC curves, precision values and recall values generated by RF, GBC and SVM were illustrated.