# The Effect of Machine Learning Algorithms on Metagenomics Gene Prediction

Amani Al-Ajlan
Computer Science Department
College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia
aalajlan@ksu.edu.sa

Achraf El Allali
Computer Science Department
College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia
eachraf@gmail.com

## ABSTRACT

The development of next generation sequencing facilitates the study of metagenomics. Computational gene prediction aims to find the location of genes in a given DNA sequence. Gene prediction in metagenomics is a challenging task because of the short and fragmented nature of the data. Our previous framework minimum redundancy maximum relevance - support vector machines (mRMR-SVM) produced promising results in metagenomics gene prediction. In this paper, we review available metagenomics gene prediction programs and study the effect of the machine learning approach on gene prediction by altering the underlining machine learning algorithm in our previous framework. Overall, SVM produces the highest accuracy based on tests performed on a simulated dataset.

## CCS Concepts

**Applied computing → Bioinformatics; Computing methodologies→Machine learning approaches**

## Keywords

Gene prediction; Metagenomics; Machine learning

## 1. INTRODUCTION

Metagenomics is the study of microbial communities directly in their natural environments such as soil, water, and gut samples—a single sample can contain 10,000 species [6], [30]. It has been estimated that only a small proportion of organisms in nature can be cultured using standard cultivation methods, but metagenomics does not require isolation and lab cultivation of individual species [23], [28], [30]. Metagenomics—a relatively new, but quickly developing field—enables us to understand the varieties of microorganisms, their functions, collaboration, and advancement in a specific biological system.

Metagenomic analysis has useful applications in several areas, including biological, environmental, and clinical studies [10]. The analysis of metagenomics is used to discover new diseases and help preserve human health. For example, studying microbes in our environment can help us control diseases and introduce new strategies for diagnosis. Recent advances in next generating sequencing (NGS) technologies have facilitated and improved metagenomics research by generating high-throughput, low-cost sequencing data [6], [21], [29].

Sequencing data taken directly from the environment are called metagenomes, and the study of these data is called metagenomics [30]. Machine learning techniques play an important role in solving many metagenomics problems, such as gene prediction taxonomic assignment, and comparative metagenomics analysis [27]. Finding genes and identifying their functions is essential to understanding the environment being studied and to annotating metagenomics reads [11].

From a computational perspective, a DNA sequence can be considered as a string of characters or nucleotides: A (adenine), C (cytosine), G (guanine) and T (thymine). A gene is a substring of DNA that codes for protein [17], [22]. A sequence of three nucleotides, called a codon, codes for one amino acid, and a sequence of amino acids forms a protein [17]. In prokaryotes, a gene is a sequence of codons that begins with either of the start codons (ATG, CTG, GTG, or TTG) and ends with one of the following stop codon (TGA, TAG, or TAA) [12], [22]. Gene prediction is an important step in analyzing and annotating genomics sequences. Given the genome's DNA sequence, gene prediction algorithms find the location of coding regions in a genome [15]. Gene prediction algorithms for prokaryotes are simpler than for eukaryotes, due to high gene density and simple gene structure [22]. Gene prediction problems in genomics are well established and considered solved, and metagenomics uses the same gene prediction techniques. However, the greatest challenges for gene prediction algorithms in metagenomics are the short read-length and the incomplete and fragmented nature of the data [24], [26], [30].

Computational gene prediction methods can be classified into two classes: similarity search-based and ab-initio prediction methods [4], [18]. Similarity search-based approaches identify genes by comparing DNA sequences to previously known genes in protein databases, such as the basic local alignment search tool (BLAST) [3]. This method is reliable but computationally expensive. Moreover, similarity-based approaches are unable to predict novel genes that have no similarities to known genes. Ab-initio methods predict genes by finding patterns in sequences using statistical algorithms. Statistical features for coding regions are different from the features in non-coding regions, and ab-initio methods predict genes using intrinsic features of DNA sequences, such as sequence length, codon usage, and GC content. Codon usage is the most common feature in a range of gene prediction programs,


*ICBRA '18*, December 27–29, 2018, Hong Kong, Hong Kong

ACM ISBN 978-1-4503-6611-3/18/12…$15.00
DOI: https://doi.org/10.1145/3309129.3309136

referring to the frequency of every codon in a sequence. Ab-initio methods are capable of discovering novel genes at reasonable computational cost.

In this paper, we review existing metagenomics gene prediction methods. Then, we study the effect of different machine learning algorithms in predicting genes using our previous state-of-the-art framework and a large metagenomics dataset.

## 2. RELATED WORK

Most conventional gene prediction programs start by identifying the open reading frames (ORFs), then classifying them as coding or non-coding. An ORF is a sequence that starts with a start codon and ends with a stop codon; metagenomics data are short reads and have incomplete ORFs, making gene prediction a challenging task [30], [31]. Incomplete ORFs mean that they are missing a start codon, stop codon, or both, and some gene prediction programs fail to identify incomplete ORFs that are part of a gene [30]. There are many metagenomics gene prediction programs including Orphelia [12], [13], metagenomics gene call (MGC) [2], Prodigal [14], FragGeneScan [25], Glimmer-MG [16], MetaGene [19], MetaGeneAnnotator [20], MetaGeneMark [33] and Meta-MFDL[32].

Orphelia [12], [13] is a gene prediction algorithm that uses a two-stage machine learning approach. In the first stage, Orphelia extracts a large number of features, including codon usage, dicodon usage, and translation initiation sites, from each ORF and applies linear discriminants to reduce the feature space. In the second stage, a neural network is used to combine the features from the previous stage with the ORF length and the GC content of the fragment. The neural network computes the probability that an ORF encodes a protein and, finally, a greedy algorithm is used to select the most probable gene from the ORFs that overlap.

MGC [2] is another prediction algorithm that uses the same two-stage machine learning approach. MGC adds two features — monoamino acid usage and diamino acid usage; moreover, MGC builds several classification models for different GC content and uses the appropriate model for each fragment. MGC shows the importance of separating the classification models based on GC content by showing that the ensemble technique outperforms the original Orphelia algorithm. Comparison results show that MGC outperforms Orphelia.

mRMR-SVM [1] is another improvement over our initial MGC algorithm. We first extract complete and incomplete ORFs from each fragment, then we compute the codon usage, dicodon usage, monoamino acid usage, and diamino acid usage from each ORF. Minimum redundancy maximum relevance (mRMR) is then used to select the best 500 features. The hypothesis is that selecting the most relevant features is better than linearly combining features of the same origin as was the case of the two-machine learning approach. We then add features from the ORF length and the fragment GC content. An ensemble of support vector machines (SVM) are built for each GC content range and used to obtain gene probability for candidate ORFs. Lastly, a greedy algorithm uses the probability from the previous step to resolve any overlap in the predictions. We compare our new framework to the both original approaches and results show that the new algorithm outperforms the original two-stage machine learning approaches.

Prodigal [14] is one of the most popular gene prediction tools for bacterial and archaeal genomes. Prodigal extracts set of features from input sequences: start codon usage, ribosomal binding site (RBS) motif usage, GC frame plot bias, and the hexamer coding statistics. Dynamic programming is then used to predict genes.

FragGeneScan [25] is a gene prediction algorithm based on the hidden Markov model (HMM). It combines sequence error models, codon usage, and sequence patterns for start/stop codons, using the HMM to improve the accuracy of gene predictions. In the main module, genes are predicted after determining the path of hidden states that best generates the observed nucleotide sequence using the Viterbi algorithms. FragGeneScan has two useful features: the ability to predict fragmented genes and the ability to predict genes with frameshifts, which result from insertion or deletion sequence errors. FragGeneScan outperforms most methods in erroneous reads, however it does not perform as well as the other approaches in the absence of errors in the reads.

Glimmer-MG [16] is another gene prediction algorithm, based on interpolated Markov models (IMM), that predicts genes in erroneous sequences. Glimmer-MG uses the same model as in the original genomic gene finder and adds additional features—adjacent gene orientation, ORF length, and adjacent gene distance—to predict genes in metagenomics reads. To deal with sequencing errors, Glimmer-MG predicts insertions and deletions by branching into different frames at predefined locations. Glimmer-MG has the drawback of being computational expense as the algorithm requires classification and clustering before gene prediction each time.

MetaGene [19] is a gene prediction program for metagenomics sequences. It uses different features to predict genes, such as codon and dicodon frequencies that are estimated based on the GC content of input sequences; the frequency distribution of ORF lengths; the distance distributions from the correct start codon to the leftmost start codon; and the distances between neighboring ORFs. MetaGene uses a two-stage approach; in the first stage, all ORFs are extracted and scored based on base compositions and ORF lengths; in the second stage, dynamic programming is used to combine previous scores and the scores of orientations and distances of neighboring ORFs, to calculate an optimal combination of ORFs. The scoring scheme is based on a stochastic approach. MetaGene has two limitations: the absence of an RBS model, and less sensitivity for predicting atypical genes that have different codon usages than typical genes. The MetaGeneAnnotator [20] is an extension of MetaGene that includes statistical models for prophage genes and an RBS model—features that improve prediction accuracy. MetaGeneAnnotator has the ability to detect both typical and atypical genes and is thus more accurate than MetaGene.

MetaGeneMark [33] predicts genes in short prokaryotic sequences with unknown origin. MetaGeneMark is based on Markov models and estimates parameters from dependencies between frequencies of oligonucleotides in protein-coding regions and genome nucleotide composition.

Recently, deep learning is used to solve various bioinformatics and computational biology problems [5] including gene prediction. For example, Meta-MFDL [32] uses deep learning to predict genes in metagenomics fragments. First, ORF length coverage, monocodon usage, monoamino acid usage, and Z-curve parameter features are extracted from each ORF. Then, feature fusion is applied which concatenates the four type of features into one set in order to form a vector that represents the ORF. Then, deep stacking network is used to classify ORFs into coding or non-coding ORFs.

Table 1 compares different metagenomics gene prediction programs in terms of features and prediction approaches.

**Table 1. Metagenomics gene prediction programs**

| Program | Features | Prediction approach |
|---|---|---|
| Orphelia | mononcodon usage, dicodon usage, translation initiation sites, ORF length and GC content | linear discriminants and neural network |
| MGC | monocodon usage, dicodon usage, monoamino-acid usage, diamino-acid usage, TIS coverage, TIS probability, complete ORF length, incomplete ORF length and GC-content | linear discriminants and neural network |
| mRMR-SVM | monocodon usage, dicodon usage, monoamino-acid usage, diamino-acid usage, complete ORF length, incomplete ORF length and GC content | mRMR and SVM |
| Prodigal | start codon usage, ribosomal binding site (RBS) motif usage, GC frame plot bias, hexamer coding statistics | dynamic programming |
| FragGeneScan | sequence error models, codon usage and sequence pattern for start/stop codon | hidden Markov models |
| Glimmer-MG | ribosome binding site RBS, TIS, start codon usage, adjacent gene orientation, ORF length and adjacent gene distance. | interpolated Markov model |
| MetaGene | codon frequencies, dicodon frequencies, ORF lengths, the distance from leftmost start codons, the distances between neighboring ORFs | dynamic programming |
| MetaGeneAnnotator | codon frequencies, dicodon frequencies, ORF lengths, the distance from leftmost start codons, the distances between neighboring ORFs, models for prophage genes and RBS | dnamic programming |
| MetaGeneMark | dicodon frequencies, nucleotide frequencies | hidden Markov model |
| Meta-MFDL | ORF length coverage, monocodon usage, monoamino acid usage, and Z-curve parameter features | deep stacking network |

# 3. MATERIALS AND METHODS

## 3.1 Dataset

We use the same dataset originally used by Orphelia [13], which contains 7 million ORFs extracted from fragments of length 700 bp. These fragments were simulated from 131 bacterial and archaeal genomes, and their gene annotations was obtained from GenBank [7]. We divide the dataset into 10 mutually exclusive parts, based on fragments' GC content. Previous research [2] has shown that building several classification models based on pre-defined GC ranges improves prediction performance and outperforms a single model.

## 3.2 Methodology

In order to study to effect of the machine learning classifiers on our perdition problem, we adopt our previous framework [1] to perform the experiments. Figure 1 shows the main stages of this framework. During the classification phase, the underlining classification algorithm is altered each time in order to test different classifiers. The predictions from this phase are fed to the next and final step in order to remove any overlap in the prediction. The accuracy of each run is computed by comparing the final prediction set with the real annotations previously computed for our simulated dataset. Below is a description of all five phases:

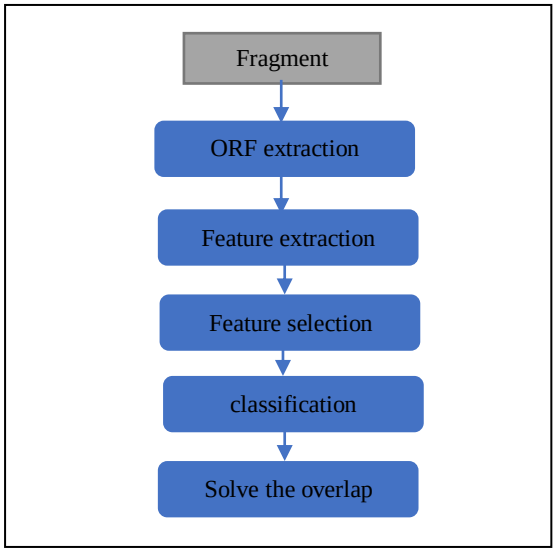


**Figure 1. mRMR-SVM framework**

**ORF extraction**: all complete and incomplete ORFs are extracted from each fragment. Since metagenomics fragments come from short NGS reads, these reads can either overlap with a gene, be completely inside a gene, or reside completely in the non-coding area. The results of this step include all possible ORFs that can be either complete, missing upstream portion, downstream portion or both.

**Feature extraction**: 4,000 features from each ORF are extracted in this step. These features include codon usage, dicodon usage, monoamino acid usage, and diamino acid usage. Codon usage represents the frequency of each codon in the given ORF while dicodon usage represents the frequency of each overlapping codon tuples in the read. Similarly, monoamino acid and diamino acid usages are extracted using the translation of the DNA sequence of the ORF into amino acid sequence. Additionally, we compute the

ratio of the ORF length to the read length as well as the GC content of the fragment.

**Feature selection:** Feature selection reduces the dimensionality of data by removing irrelevant or redundant features according to certain criteria. There are many advantages of the feature selection phase in a machine learning experiment. It facilitates data understanding, reduces memory usage, reduces training time, and increases the performance of classification. Feature selection methods can be categorized into: filter, wrapper, hybrid and embedded methods depending on how to select and evaluate features. In our framework, we use a filter feature selection method that uses intrinsic characteristics of data and evaluates features independent of learning algorithms. Filter methods are suitable for high dimensional data because they are simple, fast, and computationally efficient. In our previous framework, we selected the best filter method by choosing a classification approach and then varying the filter methods and the final number of features. The mimimum redundancy maximum relevancy (mRMR) was selected and is used here to select best 500 features from the original feature space that best characterizes the statistical properties of the ORF. These features are the most dissimilar to one another and most similar to the classification variable.

**Classification:** the ORF length feature, GC content, and best 500 features selected previously are supplied to different machine learning algorithms to obtain the probability that an ORF encodes protein. In this paper, we test five different machine learning algorithms including k-nearest neighbors (KNN), naïve Bayes (NB), random forest (RF), neural networks (NN), and support vector machines (SVM). Our goal is to study the effect of the machine learning algorithms on our particular prediction problem. For each machine learning algorithm, we build 10 classification models based on pre-defined GC ranges. The ensemble technique has been shown to outperform a single model as was proven by our previous research [2]. After tuning the parameters of each classifier using a separate dataset, cross-validation is used to evaluate the prediction performance for each algorithm. We use the modified version of each algorithm that outputs the probability that a prediction belongs to the positive class (the coding class in our case). The reason behind this is due to the fact that overlapping ORFs can be classified as coding while only one of them can be a real gene. The next and final step uses these probabilities.

**Solve the overlap:** some candidate genes might overlap or belong to the same ORF set. In this case, we use the same greedy algorithm from our previous work [1], which takes the probability from the classification stage and select the most probable ORF to include in the final set of genes.

## 4. RESULTS AND DISCUSSION

The aim of our study is to evaluate different machine learning algorithms to predict genes in metagenomics fragments. We focus on the classification phase and compare the accuracy of gene prediction framework using five different classification algorithms: k-nearest neighbors (KNN), naïve Bayes (NB), random forest (RF), neural networks (NN), and support vector machines (SVM). Tests are performed on fragments simulated from 131 bacterial and archaeal genomes and processed using our previous framework as shown in the methodology section. Previous research has shown that building multiple models based on GC content is better than building a single model [2]. Therefore, for each machine learning algorithm, we perform cross-validation on 10 subsets of the dataset divided by GC content. Given the high dimensionality of the data and the large number of models to build, we use the Amazon Elastic Compute Cloud (Amazon EC2) to perform all experiments [9].

**Table 2. Accuracy of different machine learning algorithms**

| GC Ranges | KNN | NB | RF | NN | SVM |
|---|---|---|---|---|---|
| 0-36.57 | 97.29 | 93.30 | 95.80 | 96.99 | 97.89 |
| 36.57-41.57 | 97.95 | 93.92 | 96.35 | 97.64 | 98.37 |
| 41.57-46 | 98.00 | 93.95 | 96.07 | 97.76 | 98.40 |
| 46-50.14 | 97.62 | 92.30 | 95.64 | 97.34 | 98.28 |
| 50.14-54.28 | 97.41 | 91.06 | 94.86 | 97.16 | 98.22 |
| 54.28-58.14 | 97.10 | 89.80 | 94.94 | 96.91 | 98.05 |
| 58.14-61.85 | 97.09 | 89.25 | 94.50 | 97.44 | 98.30 |
| 61.85-65 | 97.44 | 90.81 | 95.02 | 97.83 | 98.70 |
| 65-68.28 | 97.75 | 91.33 | 95.22 | 98.19 | 98.95 |
| 68.28-100 | 98.04 | 91.75 | 95.69 | 98.31 | 99.08 |
| **Average** | **97.57** | **91.75** | **95.41** | **97.56** | **98.42** |

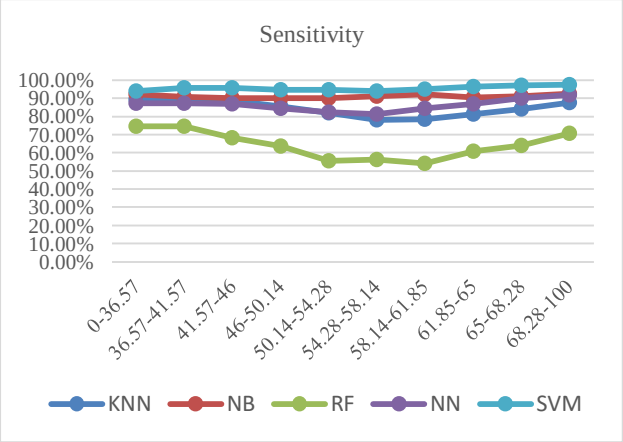


**Figure 2. Sensitivity of different machine learning algorithms**

Table 2 presents the accuracy of each method on all 10 GC ranges. The SVM classifier achieves the highest accuracy across all GC ranges—an average accuracy of 98.42%—followed by k-nearest neighbors at 97.57%. The naïve Bayes achieves the lowest accuracy, of 91.75%. Figures 2 and 3 show the sensitivity and specificity of different machine learning methods on all 10 GC ranges. The results also show the performance difference across the GC spectrum. For example, we can see that the NB and RF algorithms have lower specificity and sensitivity respectively for high GC ranges. Overall, the SVM classifier achieves the highest sensitivity in all GC ranges and random forest has the lowest sensitivity. Moreover, all machine learning algorithms have high specificity across all GC range. SVM has been successfully used in various fields, including text classification, object recognition, image processing and many more classification problems. SVM is also widely used in bioinformatics and computational biology because it is computationally efficient, robust in high dimensional

data and is less prone to over-fitting. A survey of the use of SVM in bioinformatics lists many useful applications of SVM in the field such as protein secondary structure prediction, gene function classification, cancer tissue classification, translation initiation site recognition, identification of protein functions [8]. Our experiments show that SVM is also the best choice for gene prediction in metagenomics. Previous research shows lower accuracy scores in high GC ranges due to the high number of short ORFs that exist in high GC fragments. In our experiment, we can see that SVM has a somewhat constant accuracy across all GC ranges and even slightly better results for high GC ranges.

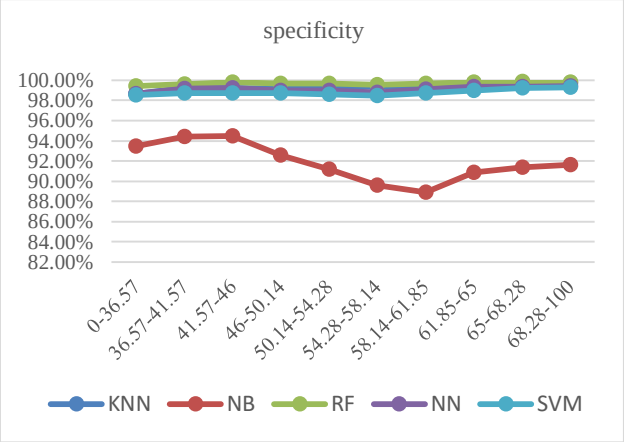


**Figure 3. Specificity of different machine learning algorithms**

## 5. CONCLUSIONS

The incomplete and fragmented nature of metagenomics data presents challenges in predicting genes in metagenomics fragments. A reliable gene prediction program is essential for improving the metagenomics pipeline and for discovering novel genes and their functions. In this study, we compare various machine learning algorithms for predicting genes in metagenomics fragments. We empirically demonstrate that SVM outperforms other machine learning algorithms when compared to other classification algorithms using the same prediction framework. Our findings concur with other research that conclude that SVM achieves high accuracy and outperforms other machine learning algorithms [8]. Our next step is to explore the possibility of using convolutional neural networks in metagenomics gene prediction. Moreover, we plan to incorporate sequence errors handling in our prediction models.

## 6. ACKNOWLEDGMENTS

This research project is supported by a grant from the "King Abdulaziz City for Science and Technology" (KACST), Saudi Arabia (Grant No. 1-17-02-001-0025).

## 7. REFERENCES

[1] Al-Ajlan, A. and El Allali, A. 2018. Feature selection for gene prediction in metagenomic fragments. *BioData Mining*. 11, 1 (2018), 9.

[2] El Allali, A. and Rose, J. R. 2013. MGC: a metagenomic gene caller. *BMC bioinformatics*. 14, 9 (2013), S6.

[3] Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990, Basic local alignment search tool. *Journal of molecular biology*. 215, 3 (1990), 403–410.

[4] Angelova, M., Kalajdziski, S. and Kocarev, L. 2010. Computational methods for gene finding in prokaryotes. *ICT Innovations*. (2010), 11–20.

[5] Angermueller, C., Pärnamaa, T., Parts, L. and Stegle, O. 2016. Deep learning for computational biology. *Molecular systems biology*. 12, 7 (2016), 878.

[6] Di Bella, J. M., Bao, Y., Gloor, G. B., Burton, J. P. and Reid, G. 2013. High throughput sequencing methods and analysis for microbiome research. *Journal of microbiological methods*. 95, 3 (2013), 401–414.

[7] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. and Sayers, E. W. 2011. GenBank. *Nucleic acids research*. 39, Database issue (2011), D32.

[8] Chicco, D. 2012. Support Vector Machines in Bioinformatics: a Survey. *Politecnico di Milano, Dipartimento di Elettronica e Informazione*. (2012).

[9] Cloud, A. E. C. 2011. Amazon web services. *Retrieved November*. 9, (2011), 2011.

[10] Handelsman, J. 2004. Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and molecular biology reviews*. 68, 4 (2004), 669–685.

[11] Hoff, K. J. 2009. The effect of sequencing errors on metagenomic gene prediction. *BMC genomics*. 10, 1 (2009), 520.

[12] Hoff, K. J., Lingner, T., Meinicke, P. and Tech, M. 2009. Orphelia: predicting genes in metagenomic sequencing reads. *Nucleic acids research*. 37, suppl_2 (2009), W101–W105.

[13] Hoff, K. J., Tech, M., Lingner, T., Daniel, R., Morgenstern, B. and Meinicke, P. 2008. Gene prediction in metagenomic fragments: a large scale machine learning approach. *BMC bioinformatics*. 9, 1 (2008), 217.

[14] Hyatt, D., Chen, G.-L., LoCascio, P. F., Land, M. L., Larimer, F. W. and Hauser, L. J. 2010. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC bioinformatics*. 11, 1 (2010), 119.

[15] Jones, N. C., Pevzner, P. A. and Pevzner, P. 2004. *An introduction to bioinformatics algorithms*. MIT press.

[16] Kelley, D. R., Liu, B., Delcher, A. L., Pop, M. and Salzberg, S.L. 2011. Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering. *Nucleic acids research*. 40, 1 (2011), e9–e9.

[17] Lewin, B., Krebs, J., Kilpatrick, S. T. and Goldstein, E. S. 2011. *Lewin's genes X*. Jones & Bartlett Learning.

[18] Mathé, C., Sagot, M., Schiex, T. and Rouzé, P. 2002. Current methods of gene prediction, their strengths and weaknesses. *Nucleic acids research*. 30, 19 (2002), 4103–4117.

[19] Noguchi, H., Park, J. and Takagi, T. 2006. MetaGene: prokaryotic gene finding from environmental genome shotgun sequences. *Nucleic acids research*. 34, 19 (2006), 5623–5630.

[20] Noguchi, H., Taniguchi, T. and Itoh, T. 2008. MetaGeneAnnotator: detecting species-specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes. *DNA research*. 15, 6 (2008), 387–396.

[21] Oulas, A., Pavloudi, C., Polymenakou, P., Pavlopoulos, G. A., Papanikolaou, N., Kotoulas, G., Arvanitidis, C. and Iliopoulos, Ioannis 2015. Metagenomics: tools and insights for analyzing next-generation sequencing data derived from biodiversity studies. *Bioinformatics and biology insights*. 9, (2015), BBI-S12462.

[22] Pérez-Rodríguez, J. and García-Pedrajas, N. 2011. An evolutionary algorithm for gene structure prediction. *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (2011), 386–395.

[23] Quince, C., Walker, A. W., Simpson, J. T., Loman, N. J. and Segata, N. 2017. Shotgun metagenomics, from sampling to analysis. *Nature biotechnology*. 35, 9 (2017), 833.

[24] Rangwala, H., Charuvaka, A. and Rasheed, Z. 2014. Machine learning approaches for metagenomics. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2014), 512–515.

[25] Rho, M., Tang, H. and Ye, Y. 2010. FragGeneScan: predicting genes in short and error-prone reads. *Nucleic acids research*. 38, 20 (2010), e191–e191.

[26] Roumpeka, D. D., Wallace, R. J., Escalettes, F., Fotheringham, I. and Watson, M. 2017. A review of bioinformatics tools for bio-prospecting from metagenomic sequence data. *Frontiers in genetics*. 8, (2017), 23.

[27] Soueidan, H. and Nikolski, M. 2017. Machine learning for metagenomics: methods and tools. *Metagenomics*. 1, 1 (2017).

[28] Thomas, T., Gilbert, J. and Meyer, F. 2012. Metagenomics-a guide from sampling to data analysis. *Microbial informatics and experimentation*. 2, 1 (2012), 3.

[29] Wang, Z., Chen, Y. and Li, Y. 2004. A brief review of computational gene prediction methods. *Genomics, proteomics & bioinformatics*. 2, 4 (2004), 216–221.

[30] Wooley, J. C., Godzik, A. and Friedberg, I. 2010. A primer on metagenomics. *PLoS computational biology*. 6, 2 (2010), e1000667.

[31] Yok, N. and Rosen, G. 2010. Benchmarking of gene prediction programs for metagenomic data. *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE* (2010), 6190–6193.

[32] Zhang, S.-W., Jin, X.-Y. and Zhang, T. 2017. Gene Prediction in Metagenomic Fragments with Deep Learning. *BioMed research international*. 2017, (2017).

[33] Zhu, W., Lomsadze, A. and Borodovsky, M. 2010. Ab initio gene identification in metagenomic sequences. *Nucleic acids research*. 38, 12 (2010), e132–e132.